ESPOO 2004 VTT PUBLICATIONS 535

methods, processes, best practices, softwar
SPI
uality, success factors, management, commitm

Seija Komi-Sirviö

Development and Evaluation of Software Process Improvement Methods

VTT PUBLICATIONS 535

Development and Evaluation of Software Process Improvement Methods

Seija Komi-Sirviö

VTT Electronics

Academic Dissertation to be presented with the assent of the Faculty of Science, University of Oulu, for public discussion in the Auditorium YB210, Linnanmaa, on June 18th, 2004, at 12 noon.

ISBN 951–38–6388–3 (soft back ed.)
ISSN 1235–0621 (soft back ed.)

ISBN 951–38–6389–1 (URL: http://www.vtt.fi/inf/pdf/)
ISSN 1455–0849 (URL: http://www.vtt.fi/inf/pdf/)

Copyright © VTT Technical Research Centre of Finland 2004

JULKAISIJA – UTGIVARE – PUBLISHER

VTT, Vuorimiehentie 5, PL 2000, 02044 VTT
puh. vaihde (09) 4561, faksi (09) 456 4374

VTT, Bergsmansvägen 5, PB 2000, 02044 VTT
tel. växel (09) 4561, fax (09) 456 4374

VTT Technical Research Centre of Finland,
Vuorimiehentie 5, P.O.Box 2000, FIN–02044 VTT, Finland
phone internat. + 358 9 4561, fax + 358 9 456 4374

VTT Elektroniikka, Kaitoväylä 1, PL 1100, 90571 OULU
puh. vaihde (08) 551 2111, faksi (08) 551 2320

VTT Elektronik, Kaitoväylä 1, PB 1100, 90571 ULEÅBORG
tel. växel (08) 551 2111, fax (08) 551 2320

VTT Electronics, Kaitoväylä 1, P.O.Box 1100, FIN–90571 OULU, Finland
phone internat. + 358 8 551 2111, fax + 358 8 551 2320

Technical editing Maini Manninen

Otamedia Oy, Espoo 2004

Komi-Sirviö, Seija. Development and Evaluation of Software Process Improvement Methods. Espoo 2004. VTT Publications 535. 175 p. + app. 78 p.

Keywords software process improvement (SPI), SPI measurement, software quality, knowledge management, Pr^{2}imer, Profes

Abstract

Software development is in constant change. New software development strategies, methods, processes, and tools are constantly introduced and taken in use. Simultaneously, the growth and importance of software has accelerated, and software has become a fundamental part of a whole range of different products. Software development strategies are changing as well: globally distributed software development, use of commercial off-the-shelf (COTS), and Open Source development are some examples of the latest tendencies. Ever-tightening competition has led to shortened lead-time requirements and variety of customised software versions targeted to divergent markets. Software development needs to be optimised to meet these challenges - without sacrificing quality. To keep abreast of change software process improvement (SPI) should develop, too, over time.

Well-managed software development processes has become strategic core competency in many organisations, enabling high-class software development, quality estimation, control, and prediction. However, improving software development processes is demanding and complex task. Numerous software process improvement (SPI) methods in the market offer help and guidance, but unfortunately they only partially address factors found essential for achieving SPI success.

This dissertation develops, presents and argues for the SPI methods embodying characteristics directing towards successful process improvement. As the results, the thesis extracts critical success factors for SPI initiatives using SPI lessons learnt. Furthermore, it incrementally develops and evaluates SPI methods, incorporating means to achieve the above-mentioned critical success factors. The research is based on several industrial case studies.

Preface

This thesis is based on the software process research I was involved in at VTT Electronics during the period 1994 through 2003. My interest towards software and software process quality was raised as early as 1991 while working on my master’s thesis on Total Quality Control at CCC companies. Later, at VTT Electronics, I was able to continue the quality related research and carry out a series of Tekes and EU funded research projects focusing on software process improvement and measurement.

I wish to express my sincerest gratitude to Professor Veikko Seppänen, the supervisor of this thesis at the University of Oulu and my former superior, who has greatly encouraged me through all these years to continue the research and to finalise it in the form of a doctoral thesis. I cannot thank him enough for all the guidance and valuable comments he has given on the several drafts of this thesis. The reviewers of the thesis, Professor Victor Basili and Professor Jyrki Kontio deserve my sincere thanks for their time and effort they have spent in reviewing my research and giving their profound comments, which have helped me to improve the quality of the thesis dramatically.

I wish to give my special thanks to my former superior Professor Markku Oivo for his expert comments on the work in hand. Dr. Ioana Rus deserves a special mention for her valuable comments and continuous encouragement during the final stages of the writing process of this thesis.

During the long course of the research I have had countless fruitful discussions and debates with my colleagues and friends. I wish to express my sincere thanks to all VTT colleagues in general and my research fellows in the embedded software engineering research group in particular, with whom I have had an opportunity to enjoy and carry on this research. The period of research has been one decade which makes it difficult to thank individually all those colleagues and friends home and abroad who have influenced and contributed the work at hand; thus I would like to thank anonymously all of them.

Without the industrial interest and involvement this research would not have been possible. ABB Transmit Oy Relays and Network Control, Datex Ohmeda Finland (former Datex Instrumentarium), Dräger Medical Technology, Instrumentarium Imaging, LM Ericsson, Nokia Networks, Nokia Paging,

Tokheim (former Schlumberger Retail Petroleum Systems), and Valmet Automation deserve my sincere thanks for enabling this research by providing a challenging software development environment for developing, testing and further developing the results of this research in various stages. Their feedback and co-operation has been vital to this research and deserves the deepest gratitude.

I would like to express my sincere gratitude to VTT Electronics for providing me with the opportunity and resources needed to carry out and finalise the research at hand. In addition, I have also received financial support from the following foundations: Tauno Tönning säätiö, Seppo Säynäjäkankaan rahasto and Tekniikan säätiö. This support is gratefully acknowledged.

I also wish to thank Mr. Seppo Keränen for checking the language of this thesis.

Finally, I wish to express my sincere gratefulness to my beloved ones for their loving support and understanding. My mother has always believed in me and encouraged me to proceed with my studies. Ever since I began school I heard how important it is to study. My beloved children Ellinoora and Ruupe, my endless source of joy and inspiration, gave me the best reason to finalise this research, for then I would be able to spend more time with them. My husband Pekka has been my encourager during all these years. He has shown endless understanding of my evening and weekend working periods by unselfishly providing me the possibility to concentrate on the research. His love, support and understanding made this work at large possible.

Kiiminki, April 2004-04-25 Seija Komi-Sirviö

List of Original Papers

This thesis includes the following original papers published in the proceedings of international conferences and journals (Appendices I–VII).

I Karjalainen, J., Mäkäräinen, M., Komi-Sirviö, S. & Seppänen, V. 1996. Practical process improvement for embedded real-time software, Quality Engineering. Vol. 8, No. 4. Pp. 565–573.

II Komi-Sirviö, S., Oivo, M. & Seppänen, V. 1998. Experiences from practical software process improvement. In the Proceeding of EuroSPI'98. Gothenburg, Institutet för verkstadsteknisk forskning IVF. November 16–18, 1998. Pp. 5.31–5.45.

III Parviainen, P., Komi-Sirviö, S. & Sandelin, T. 1998. Measurement-based improvement of critical software subprocesses: Experiences from two industrial cases. In the Proceedings of European conference of Software Process Improvement (SPI'98). Monaco, December 1–4. 10 p.

IV Komi-Sirviö, S., Parviainen, P. & Ronkainen, J. 2001. Measurement Automation: Methodological Background and Practical Solutions – A Multiple Case Study. In the proceedings of the 7th International Software Metrics Symposium, IEEE Computer Society. London, April 4–6, 2001. Pp. 306–316.

V Birk, A., Järvinen, J., Komi-Sirviö, S., Kuvaja, P., Oivo, M. & Pfahl, D. 1998. Profes - A product driven process improvement methodology. In the Proceedings of European Conference on Software Process Improvement (SPI'98). Monaco, December 1–4, 1998. 9 p.

VI Järvinen, J., Komi-Sirviö, S. & Ruhe, G. 2000. The Profes Improvement Methodology - Enabling Technologies and Methodology Design. In the Proceedings of the Second International Conference on Product Focused Software Process Improvement (Profes 2000). Oulu, June 20–22. Pp. 257–270.

VII Komi-Sirviö, S., Mäntyniemi A. & Seppänen V. 2002. Toward a Practical Solution for Capturing Knowledge for Software Projects, IEEE Software May/June 2002. Pp. 60–62.

The papers will be referred to in the text by the corresponding Roman numerals.

Paper I introduces the main ideas, principles, and basic functionality of Pr^{2}imer, the practical process improvement method, which, as an SPI method, unites various SPI approaches, e.g., process analysis, measurement and process modelling, to a functional ensemble. Paper II summarises the experiences and lessons learnt that have been gained from applying the Pr^{2}imer method to SPI in the course of five years. Paper III reinforces Pr^{2}imer with strengthened goal driven measurement activities and applies the enhanced method to testing and requirements engineering processes. Furthermore, Paper IV illustrates how measurement automation can be utilised to facilitate SPI and management processes. Paper V describes the change in SPI strategy. The paper presents how to move the improvement basis beyond processes, to the quality of the end product. It introduces Profes, the product quality driven improvement methodology, which supplements the Pr^{2}imer method with a product focus. Paper VI recapitulates the status of the Profes improvement methodology and design rationale. Paper VII is concerned with the lesson learnt and the necessity of a need based approach in the area of knowledge management driven SPI.

The author of this dissertation is the principal author of papers II, IV, V, and VII and wrote papers I, III, and VI together with the co-authors. In paper I, the author was responsible for putting the improvement cycle together as a comprehensive method. In papers III and VI the author was involved in the design of the key methodological elements.

Contents

Papers I–VII of this publication are not included in the PDF version. Please order the printed version to get the complete publication (http://www.vtt.fi/inf/pdf/)

List of Abbreviations

BSC	Balanced Scorecard
CMMI	Capability Maturity Model Integrated
CSF	Critical Success Factor
DoD	Department of Defence
ESPRIT	European Strategic Program for Research and development in Information Technology
EF	Experience Factory
EFQM	European Foundation for Quality Management (http://www.efqm.org)
GQM	Goal/Question/Metric paradigm (e.g. http://www.gqm.nl)
IEEE	Institute of Electrical and Electronics Engineers (http://www.ieee.org)
ISO	The International Standardization Organization (http://www.iso.ch)
KM	Knowledge Management
NASA	National Air and Space Agency
PPD	Product-Process Dependency
Pr^{2}imer	Practical Process Improvement for Embedded Real-Time Software
PROFES	Product Focused Software Process Improvement methodology
PSM	Practical Software Measurement
QIP	Quality Improvement Paradigm
SEI	The Software Engineering Institute
SE-CMM	Systems Capability Maturity Model
SPC	Statistical Process Control
SPI	Software Process Improvement
SPICE	Software Process Improvement and Capability determination (http://www.sqi.gu.edu.au/spice/)
SW-CMM	Software Capability Maturity Model (http://www.sei.cmu.edu/cmm)
TQC	Total Quality Control
TQM	Total Quality Management
TTM	Time to Market
VTT	Technical Research Centre of Finland (http://www.vtt.fi)

1. Introduction

The introduction summarises the background to the research, puts forward the research setting and describes the author's contribution to the research.

1.1 Background

The purpose of this chapter is to lay the background for SPI by discussing the role of software, the changes in software development strategies and software development process models, and the quality movement. Furthermore, SPI experiences are discussed.

1.1.1 The Role of Software

Software is everywhere. In the eighties, it spread out from mainframes to PC's. Today it is hidden in the products we use daily, such as home appliances, mobile phones and vehicles. Software controls vital functions in operating theatre; it keeps aeroplanes in the sky and factories in operation. It is embedded into the houses we live in; it controls temperature and lights along with providing safety. In future, we may be wearing smart clothes; and it will be possible for rooms to adapt to our personal needs as we enter them. The economic importance of software is incontestable as well as the dependence of society on software. A concrete example of this was the Year 2000 bug along with the global fear of software systems collapsing with disastrous effects. Another example is provided by the transition to Euro that kept software developers busy all over the world for a long time.

The role of software is today more important than ever, and its importance is continually increasing. The functionality and parallel versions of products realised by software have increased greatly along with the rising significance of software quality. Software has become a strategic core technology and an inseparable part of many systems (cf., e.g. Seppänen et al. 1996). The amount and complexity of software have also increased enormously, while at present the functionality and customisation of many systems is often realised by software. In addition, market requirements have led to tightened lead-time requirements, i.e., software should be produced more and more rapidly. Unfortunately, this combination can cause a great amount of quality problems.

1.1.2 Software Development Strategies

The world of software development is changing radically with accelerated speed (Wang & King 2000). In addition to technological and methodological changes, the business strategies of software development have been changing and are changing remarkably as well. In the 1980's, software development was exclusively in-house activity, and no commercial off-the-shelf (COTS) were known (Niemelä et al. 2000), not to mention the Open Source development approach (Fuggetta 2003), which is one of the most recent strategies used for enhancing software development. Through these changes, companies have been seeking, e.g., to enhance the ways of developing software, to improve quality, or to strengthen their competitive position. Another example of the latest tendencies in software development is the building of virtual organisations, in which partners may be sought separately for each project (Mowshowitz 1997, Rahikkala 2000). Currently software development is often multi-site operation, software is either developed within one organisation but by different sites spread out geographically or by co-operation with other companies using, e.g., a subcontracting schema. According to (Nasscom 2000), 185 major software development companies out of 500 had outsourced software development to India, the yearly growing rate being 53% at present.

1.1.3 Software Development Process Models

Software development process models guide software development. Trying to keep abreast with technological development and new development strategies, software development process models have gone through a long path starting in 1970, when the first software development model called the Waterfall model was introduced by Royce (1970). The basic idea was that software development should progress from phase to phase, each producing outputs that are used as inputs in the next phase. The model defined phases, activities and outputs that should be completed in strict order. Since then various software development models like Iterative Enhancement (Basili & Turner 1975), Incremental (Mills et al. 1980), Evolutionary (Gilb 1988), Prototype (Curtis et al. 1987), Spiral (Boehm 1988), or V-model (GMOD, 1992) have been introduced. These models try to improve the demerits of existing ones and/or to answer the new challenges software development has faced. The creation of new software development models has not stopped; the latest arrivals are called Agile methods (Abrahamsson et al. 2002). One well-known Agile method currently gaining a lot of interest among practitioners and in the software research community is Extreme programming (Beck 1999). Further representatives of agile methods, often called light software development methods are, e.g., the Scrum (Schwaber & Beedle 2002) and Crystal methods (Cockburn 2001). These methods are often utilised in developing applications rapidly and incrementally in situations where the time to market is a crucial driver for development.

1.1.4 The Quest for Improvement

The purpose of improvement is often to enhance software development in order to raise the quality of software (Basili & Galdiera 1995). On the other hand, the goal may be to shorten the delivery cycle, to lower the costs and thus improve profitability, or to strengthen the market position (Herbsleb et al. 1994). There may also be a need to prove the maturity of development (Humphrey & Sweet 1987), which may require changes in software development processes.

1.1.4.1 Quality

It is difficult to define the term quality unambiguously; it is a matter of opinion and viewpoint. ISO 9000 defines quality as a "degree to which a set of inherent characteristics fulfils requirements" and requirement as a "need or expectation that is stated, generally implied or obligatory" (ISO 9000 2000, p. 23). Characteristic is defined as "distinguishing feature", which can be inherent or assigned, qualitative or quantitative and it can have various classes (ISO 9000 2000, p. 31). When defining Total Quality Control Ishikawa (1985) approaches quality strictly from the customer point of view, the goal being to satisfy the requirements of customers. Juran (1999) defines this customer-based quality as fitness for use. It is generally accepted that competitiveness largely depends on the quality of products or services ultimately evaluated by customers.

When interpreting quality more broadly Ishikawa lists further quality aspects such as “quality of work, quality of service, quality of information, quality of process, quality of division, quality of people, including workers, engineers, managers, and executives, quality of system, quality of company, quality of objectives, etc." (Ishikawa 1985, p. 45). Accordingly, quality cannot be added later to the product, neither in the context of manufacturing (Ishikawa 1985) nor in software development, as also pointed out by Humphrey (1989).

The quality of software is often measured in the form of "ilities". Some examples of these quality attributes are reliability, usability, maintainability, portability, scalability, availability, and testability (McConnell 2002). Some of these quality attributes are more visible to customers, like reliability or usability, while others are more important for software development, e.g., maintainability. This list of "ilities", or features may be augmented with further quality criteria such as security or time to market (Offutt 2002).

The definitions of quality models for achieving, assessing or predicting software quality attributes are still under study. For example, the definition of a quality model for high-dependability incorporating many of these quality attributes is currently being prepared. In 2001 NASA launched a 5-year project called HDCP (High Dependability Computing Project, http://www.hdcp.org/) aiming to develop and experiment high-dependability quality models, techniques and tools to be applied to mission software.

1.1.4.2 The Quality Movement

The quality movement started in Japan as early as the late 1940's. In 1949 the Union of Japanese Scientist and Engineers organised the Quality Control Research Group and started a nation-wide quality-training project (Noguchi 1995). In 1982, Deming, one of the main opinion leaders in the Japanese quality movement, introduced Plan-Do-Check-Act, the improvement oriented management model (Deming 1982). The PDCA circle also known as the control circle was later applied to software development as well.

The quality movement entered into software development rather late, as it was only in the mid 80's that practitioners and researchers started to become interested in developing SPI models and approaches. Up to this date this development has continued producing various new and enhanced models, which are sometimes even conceived as competitors to one another. Since the introduction of Software Capability Maturity Model (CMM, later known and referred using SW-CMM abbreviation) (Humphrey & Sweet 1987) the quality of the software development process has been widely addressed. Thereafter a variety of process maturity models using similar principles have been announced such as Bootstrap (Kuvaja & Bicego 1993, Kuvaja et al. 1994) and ISO 15504 (El Emam et al. 1998). In addition to the capability or maturity based process quality models, numerous process standards (for example ISO 9000 standard series or IEEE standards) and application area specific standards have put forward examples of quality software process. In addition to these models, there are some further improvement instruments such as the ISO 9000 certification, the European Quality Award (EFQM 2003) and Malcolm Baldrige Award (BNQP 2003) that are used by some software developing organisations today.

For the sake of competitiveness, companies have invested a lot of resources in improving software quality. However, the success has not been straightforward or easy to achieve. To facilitate SPI in Europe, the European Systems and Software Initiative (ESSI) was established and executed by the European Commission during 1998–2000. Under this initiative, more than 470 projects in the area of software and systems improvement were funded (Haug et al. 2001).

1.1.5 SPI Experiences

SPI denotes activities aiming at improving the software development process and is used for reaching a desired improvement goal. SPI methods are the instruments used for guiding and managing improvement activities in practice.

1.1.5.1 Success Stories

Many successful SPI studies have been published in the literature. For example, Fitzgerald and O'Kane (1999) have reported how the Motorola Cellular Infrastructure group has successfully reached SW-CMM level 4 and is heading for level 5. Motorola had also previously reported good results using the SW-CMM-based process improvement method. They have even calculated the total return of investment to have soared to 677% when raising their capability from SW-CMM level 2 to 5. This calculation is based on required SPI investment, rework originating from defects, and differences in defect rates between the SW-CMM level 2 and level 5 projects (Diaz & Sligo 1997). L. M. Ericsson has described their successful SPI initiatives in the terms of quality, delivery precision and lead times (Dyne 1998). The Software Engineering Institute collected data from 13 organisations to analyse SW-CMM-based SPI results (Herbsleb et al. 1994). According to this analysis, the productivity improved per year varied from 9% to 67% (median 35%), the yearly reduction in time to market improved from 15% to 23%, and the Return of Investment (ROI) was calculated to have raised from 4.0 to 8.8. Detailed examples of the benefits were reported as follows: defects/KSLOC dropped from .4 to .11 (Hewlett Packard), product planning slippage reduced during 1990–1992 from 50% to 5% (Schlumberger), every invested $1 showed a $7.80 avoidance of rework costs (Raytheon) and the find and fix time for defects reduced from 8 hours each to 11 minutes (Texas Instruments).

1.1.5.2 SW-CMM Reversals

Although there are a number of published SPI success stories where, e.g., the SW-CMM assessment method has been applied in SPI, the extensive survey done by SEI in 1995 put forward a slightly different scene (Goldenson & Herbsleb 1995). The SEI survey studied a total of 138 completed questionnaires from 56 appraisals from the United States and Canada; the assessment was conducted during the years 1992–1993. 75% of the respondents stated that the assessment was worth the money and effort spent and that it had had a major positive effect on the organisation. However, at the same time 26 % claimed that nothing much had changed since the assessment and even 79% reported that the process improvement was overcome by various events and crises – other things had taken priority. Based on the survey results, the authors admit that it is not clear how to proceed after the appraisal: “We need to learn more about how to make change happen, not just what needs to be improved” (Goldenson & Herbsleb 1995, p. 9). Even this can be questioned, since, based on assessment results, it might not yet be certain enough what would be wise to improve and what should be improved first.

1.1.5.3 ISO 15504 Reversals

El Emam and others (1999) have applied the same question schema as Goldenson and Herbsleb and present their results from the 14 Spice (ISO 15504) trial projects. They surveyed 14 Spice assessment based SPI cases where the assessment had been conducted earlier than 30 weeks before the survey took place. The results were saddening: in most cases not much had changed since the assessment (72%), almost two thirds of the respondents shared the opinion that the assessment was not worth the money and effort spent, and that it had had no major positive effect on the organisation. Furthermore, the cost of the process improvement effort surprised more than half of respondents (54%). There may be several reasons for the failure, the most obvious ones being those listed in the survey. The results state that in 79% of the cases other things than SPI were regarded as more important, and that SPI was overcome by various events and crises. The Spice survey results show that companies were also struggling with resource problems, only one case out of 14 was not suffering from time or resource limitations. The speed of results was another disappointment; the process improvement effort had taken longer than expected according 85% of the respondents.

1.1.5.4 European Experiences of SPI Failures

The VASIE database (Value Added Software Information for Europe) established in 1998, supported by the European Union and maintained by the European Software Institute (ESI), provides SPI data for over 250 process improvement projects. Using this publicly available data, it has been calculated that over one third of Norwegian and Swedish Process Improvement Initiatives (PIE) supported by the European Union ESSI program were reported to have failed (Conradi and Fuggetta 2002). In Finland, the results of PIE projects have been similar to these. Out of 8 projects 3 had achieved their improvement goals, 1 failing to reach them and the results of 4 projects not being observable or visible at the time the PIE project ended (VASIE 2003). Kautz & Nielsen (2000) have recently reported failed SPI case studies, which is not commonly done by researchers. A large study on the utilisation of software best practices in European companies concluded the research results as follows: “For European organizations, the message is clear: we need to be more aware of best practices and process-improvement techniques. The European software industry lags far behind the US in both awareness and application of software process improvement”(Dutta et al. 1999, p. 89).

1.1.6 Summary

While there is a universal concern about the improvement of software development, at the same time there is also a lot of uncertainty about the best means of carrying out this task. Despite all the effort given to SPI and quality research since the 1980's, many questions still remain unresolved. In addition to the SPI problems discussed above, the link between maturity-based SPI success and business success seems to be faint. For example, Motorola (Daskalantonakis 1994, Diaz & Sligo 1997) and Ericsson (Heijstek 1998, Bang 2000, Linders 2001) have announced to have high SW-CMM ratings, while Nokia has not reported any process maturity levels of the kind. Software developing organisations operate in a dynamic market with competing products, thus subjected to tight constraints concerning schedule and cost. A fundamental factor for success is the capability to focus on the right processes in improvement lest resources be wasted for nothing. A summary illustration of the ever changing and complex environment of software development and SPI is shown in *Figure 1*.

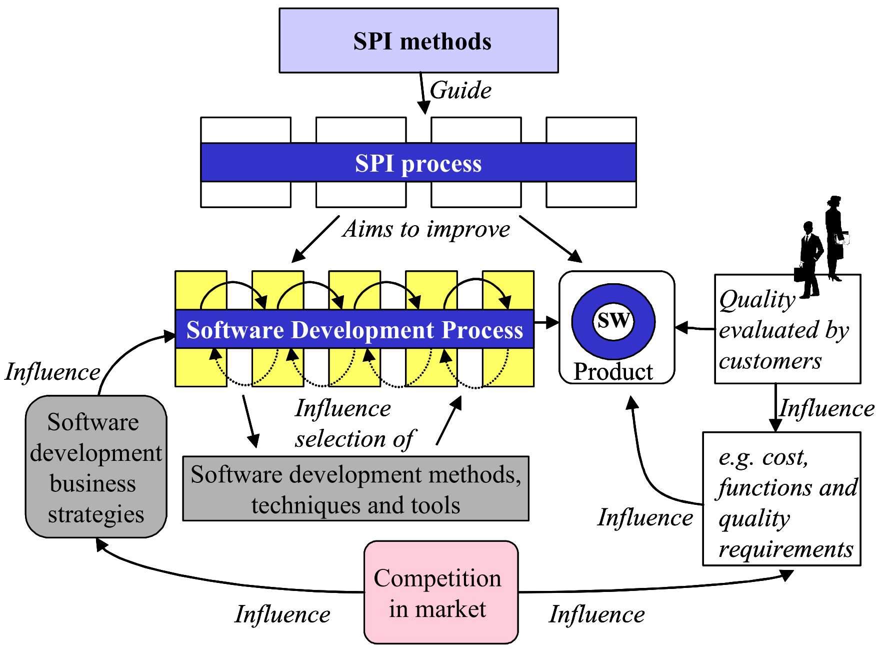
SPI methods
Guide
SPI process
Aims to improve
Software Development Process
SW
Product
Quality evaluated by customers
Influence
Software development business strategies
Influence selection of
Software development methods, techniques and tools
Influence
e.g. cost, functions and quality requirements
Influence
Competition in market
Influence
Influence

Figure 1. The environment of software process development and process improvement.

1.2 Research Setting

This chapter introduces the research problem and related questions. It also discusses the nature and scope of the research and gives an overview of the research process and methods.

1.2.1 Research Problem

SPI has been a problematic mission to carry out successfully, as already discussed and also stated by many other researchers (e.g. Ould 1996, Kasse and McQuaid 1998, Kinnula 1999). By nature, software development is human-based and complex. Software engineering has features that cannot be planned or controlled similarly to some other fields of engineering. It is, at the same time, an intellectual and a sociological design activity carried out in an environment of learning (Ould 1996). The intangible and complex nature of software engineering makes planning and controlling difficult. Reel (1999, p. 19) has determined the situation as follows: "Software systems are exceptionally complex. In fact, many agree that the basic problem of computing is the mastery of complexity. Because software developers must deal with complex problems, they are generally very intelligent and complex individuals, which also complicates the management formula". Process improvement involves making changes to current situation, which is hardly ever easy and tends to get even more difficult when it influences software specialist with high self-respect.

In the context of embedded systems, the share of software-related development work is currently often more than half of the development of the whole system. Alcatel has even estimated the software cost for switching systems to be approx. 80% of the overall cost (Debou et al. 1999); 20 years ago this product was implemented only with hardware solutions. Telecommunication organisations like Ericsson, Alcatel or Nokia once known as electronics-oriented companies now admit that their further success is solely determined by the capability to make business out of software (Dyne 1998, Debou et al. 1999). Making transformations in core engineering activities is not an easy task; to be successful, these transformations require changes in attitudes, software development models, methods and tools, work procedures, and in project management.

Continuous changes in business goals or strategies and in software development augmented with quality goals or tightened TTM requirements require changes in the ways of action as well. By improving the software processes companies seek to be more competitive and productive. While SPI is necessary for companies, it also presents challenges – and risks. And finally, the problem remains: how to execute SPI to good effect?

1.2.2 Research Questions

The continuous development of SPI methods and approaches within the ever-changing environment of software development raises the question of compatibility between developed methods and SPI needs. Thus, the research questions can be derived from the discussion above as follows:

How to develop and evaluate industrial SPI methods?

This leads to the following research questions:

Q1. What are the most typical industrial SPI needs regarding SPI methods?

Q2. What kinds of SPI methods are suited to these needs?

Q3. How to gather and analyse the practical experiences of SPI methods in order to develop them further?

The fundamental assumption of this research is that the quality of software depends on the characteristics of the processes used for producing the software as stated by (Humphrey 1989). This places great demands on the quality of the software development process. From the viewpoint of research, it is believed that SPI methods both should and can be used rationally in SPI. It has been stated in a recent research on information system development methods (Tolvanen 1998) that these methods are not to be considered as finished products but rather as evolving continuously, as dictated by technical evolution and business needs, or current information system development at hand. Consequently, the underlying proposition of this research is that the development of SPI methods is evolutionary by nature as well.

This thesis defines an SPI method as a predefined set of steps designed to guide the improvement work towards an improvement goal or goals selected by a software development project or an organisation. An SPI method may also include detailed descriptions of either recommended or needed techniques (such as brainstorming or process analysis), resources (such as tools or persons) or knowledge (such as needed expertise) on how to conduct an SPI initiative and how to implement an SPI infrastructure in an organisation. However, the definition of an SPI method in this research is closer to the dictionary definition: "the procedure of obtaining an object" (Baskerville 1996).

1.2.3 Scope of the Research

ISO 9000 defines **process** as a "set of interrelated or interacting activities which transforms inputs into outputs" (ISO 9000 2000, p. 21). The ISO Standard further defines that product is the "result of a set of interrelated activities which transforms inputs into outputs" (ISO 9000 2000, p. 23). ISO 15504 expands the definition of software process as follows: "the process or set of processes used by an organisation or project to plan, manage, execute, monitor, control and improve its software related activities" (ISO 15504-9, 1998, p. 5). It further defines **process improvement** as an "action taken to change an organisation's processes, so that they meet the organisation's business needs and achieve its business goals more effectively" (ISO 15504-9 1998, p. 5).

The scope of this research is relatively extensive: process improvement in the context of software development. While software development methods and techniques themselves fall out of the scope of this research, and thus are not dealt with, the interfaces and interaction between software engineering and SPI are included in the study.

1.2.4 Nature of the Research

According to the OECD research characterisation, dated in 1966, research and development can be divided into basic or fundamental research, applied research, and development (adapted from Niiniluoto 1993, Sintonen 1990). Basic research boosts scientific knowledge, and searches for knowledge for its own sake. Although applied research seeks knowledge accretion, too, the general aim is to "put to use the findings of basic research or even to discover new knowledge which might have immediate practical application" (Sintonen 1990, p. 24). Using these definitions, this research falls into the category of basic research.

The aim of this constructive research is to develop and to trial SPI methods in software development so as to make SPI more successful. The research strategy is to gather the relevant knowledge and to put it to use in the development and evaluation of SPI methods.

1.2.5 An Overview of the Research Process and Methods

Due to the characteristics of this research, the methods used are variform. The aim has been to develop solutions on the basis of the observed industrial problems and needs. The author has worked as a change agent in the case organisations trying to gain an understanding of the environment and projects of the organisation and its improvement needs in order to be able to propose changes and analyse the results from the viewpoints of SPI initiative success and SPI method development. Hence the action research method, as accumulated by Järvinen (1999), provides the primary research method used in this study. Rapoport (1970) was among the first to expand the change process in action research to five cycle steps (diagnosis of a problem, examination of options to solve the problem, selection of an option and execution, analysis of the results and identification of findings). However, the emphasis in this cycle is on options selection, though the approach fails to provide an adequate enough framework to support SPI method development. Later on, Adrion (1993) has put forward four further methods, one of them being the engineering method based on an evolutionary paradigm. This engineering research method provides a viable approach for SPI method development research, due to its in-built idea of continuous evolution and improvement. The engineering research method comprises the following steps (Adrion 1993, Glass 1994):

Step 1. Observe existing solutions,
Step 2. Propose better solution,
Step 3. Build or develop,
Step 4. Measure and analyse, and
Step 5. Repeat until no further improvements are possible.

This engineering research cycle was repeated four times in total. In practice, the steps 2 and 3 were joined to a single step. This was because, in the context of SPI method development, proposing better solutions was a natural part of building and developing activities, and furthermore, steps 2 and 3 were implemented iteratively during the method development cycles.

The engineering cycles were executed using several practical techniques such as interviews, teamwork, brainstorming sessions, co-writing, reviews, etc. In the third engineering research cycle also external SPI expert opinions were requested regarding the method under development.

The case companies of the research projects were selected mainly from among embedded software development companies. The only selection factors were a common interest towards SPI along with the readiness and willingness to invest in it as well.

1.2.5.1 Observe Existing Solutions

In all the engineering research cycles, a literature survey was conducted to study existing solutions and the evidence of their usefulness. The purpose was to critically analyse existing methods and to compare these methods and results with the industrial needs so as to identify any improvement areas.

1.2.5.2 Propose and Develop a Better Solution

In the first engineering research cycle, the developed answer was largely based on the best guess on the basis of the literature survey and the initial ideas of shortcomings regarding the existing methods. These ideas were processed in several brainstorming sessions, for example, and reviewed with other researchers and organisation representatives before formalising them in a proposed improvement method. In the second engineering research cycle, the enhancements of the improvement method were based on the experiences gathered and lessons learnt from the first engineering research cycle. In the third engineering research cycle, the specification of the method was composed on the basis of the general requirements set for the project, the analysis of existing improvement approaches, and industrial needs. The organisations involved in this phase stated several requirements they had set for the improvement method.

The third engineering research cycle involved two replicated method development cycles. In the first sub-cycle, the overall structure and dynamics of the methodology were formed and tested in case environments. After this, in the second sub-cycle, the method was fine-turned with more accurate and enhanced elements.

1.2.5.3 Measure and Analyse

In all engineering research cycles, the progress and intermediate results from method use were continually analysed. The analysis and follow-up of progress were based on the data collected as defined up front. The data was analysed and evaluated together with researchers, software engineers and managers. The data was used in two ways: in fine-tuning the changed software development practices and as feedback to method development.

1.2.5.4 Repeat until no Further Improvements are Possible

The steps of the engineering research method as described above were repeated four times in total. Unfortunately, in the dynamic area of software engineering it will not be conceivable to state that further SPI method enhancement would not be possible.

1.2.5.5 Development of Critical Success Factor Criteria

Critical success factor (CSF) criteria for evaluating SPI methods were also developed as a part of this research. The criteria were developed by analysing and categorising SPI success factors as presented in the literature. Here, the grounded theory research method by Glaser & Strauss (Bryant 2002, Heath & Cowley 2004) was applied in formulating the theory of SPI success factors on the ground of collected data.

1.2.6 Summary

In *Table 1* the research process and the steps taken are summarised.

Table 1. Summary of the research process.

The Engineering Research Process	**Research cycles**			
	The 1st cycle 1994–1995	The 2nd cycle 1997–1998	The 3rd cycle 1997–1999	The 4th cycle 2000–
Observe existing solutions	Including:			
	TQC, TQM GQM, Process Modelling ISO 9000 series Bootstrap SW-CMM Trillium	Pr^{2}imer SPC PSM BSC GQM	Pr^{2}imer Bootstrap GQM Ideal QIP EF ISO 15504 ISO9126	Pr^{2}imer Profes KM EF
Propose and develop better solution	Results:			
	Initial integrated SPI method *(Paper I)*	SPI method enhanced with strengthened measurement support *(Paper III & IV)*	SPI method enhanced with product quality focus *(Paper V & VI)*	SPI method enhanced with knowledge management principles *(Paper VII)*
Measure and analyse	Measurement and analysis of 3 case companies *(Paper II)*	Measurement and analysis of 3 case companies and survey of 20 companies *(Paper II)*	Measurement and analysis of 3 case companies and use of experts' opinions	*(Paper VII)*
Repeat until no further improvements are possible	Continue to the 2nd cycle	Continue to the 3rd cycle	Continue to the 4th cycle	*Work on progress*

Paper I describes the result of the first engineering research cycle, i.e., the SPI method for embedded real time software process improvement (Pr^{2}imer). Paper II discusses the experiences of using the Pr^{2}imer method. Paper III presents the Pr^{2}imer method supplemented with emphasised measurement functions and paper IV with the measurement tool support. The result of the third engineering research cycle, i.e., the description of the Profes improvement methodology is presented in paper V. Paper VI is concerned with the design rationale of Profes. Paper VII is concerned with the result of the fourth engineering research cycle, dealing with how the knowledge related to software development should be managed and used in SPI, and describing the need based approach to SPI and knowledge management.

1.3 Author's Contribution to the Research

Since 1994 the author has participated in SPI method development in four successive research projects, of which two were partly founded by Tekes, the National Technology Agency of Finland, and one by the European Commission and one by the Finnish Academy. These projects constitute the foundation of the almost ten-year long SPI research carried out by the author.

1.3.1 SPI Management and Process Quality

The first SPI method development and trial use project called ProMETRI (Komi-Sirviö 1995) was part of the larger ProHAKE program (Känsälä 1995) focused on the management and improvement of software development process. The program was started in 1994 and lasted approximately 2.5 years. The main goal of this program was to accelerate the throughput of the software development process and to enable more effective software development through selecting, applying and enhancing existing methods, instructions, and tools. The ProMETRI project was carried out by the author, the focus being on building a comprehensive and practical SPI approach using the newly developed and introduced GQM method (Basili & Rombach 1988). In this project, several industrial partners offered software development problems for practical process improvement studies. As the main result of the project, the first definition of a practical SPI method was formulated and packaged into the Pr^{2}imer approach. In addition to working as the main developer of the Pr^{2}imer method, the author was responsible for applying the method in industrial settings, including improvement planning and providing support for software development projects during the piloting period. In the course of the ProMETRI project, two industrial cases using the Pr^{2}imer method were carried out by the author. *Table 2* summarises the author's role concerning the related papers.

Table 2. Author's contribution to SPI management and process quality.

Year	Project	Author's contribution to the related papers
1994–1997	ProMETRI	Paper I: The overall Pr^{2}imer method and the case results.
		Paper II: Main author

1.3.2 Strengthened Measurement Practices

The promising experiences and new ideas led the author to plan and implement another Tekes funded process improvement program called Soihtu. The main goal of the Soihtu program was to develop, in close co-operation with companies, a set of computer-aided process modelling, assessment and measurement methods suitable for continuous improvement of the embedded software process. The program lasted two years and ended in March 1998. Within Soihtu (Soihtu 1996), the Roihu project (Roihu 1996) focused on improving software development sub-processes using the formerly developed Pr^{2}imer method. The results of this project strengthened the ideas concerning how SPI projects should be conducted and further pinpointed the importance of measurement within and after the improvement program. Measurements improved the visibility of software process, and as a result the development process become more manageable. Moreover, the influence of the improvement actions could be more easily followed up and measured. To facilitate and to uniform measurement, a measurement tool environment called MetriFlame was developed. MetriFlame supports GQM-based measurement activities through the whole measurement process, from measurement goal definition to results presentation. The Soihtu program was partially managed by the author. *Table 3* summarises the author's role concerning the related papers.

Table 3. Author's contribution to strengthened measurement practices.

Year	Project	Author's contribution to the related papers
1997–1998	Roihu	Paper II: Main author
		Paper III: The overall improvement framework and definition of how measurement relates to it
		Paper IV: Main author

1.3.3 Product Quality Focus

SPI method development was continued in the Profes project, which was financially supported by the European Commission. The project started at the beginning of 1997 and continued until September 1999. The goal of Profes was to formulate a product quality based SPI methodology using and enhancing existing process improvement approaches. The basic functionality and dynamics of Pr^{2}imer supported with the QIP principles (Basili et al. 1994b) formed the basis for Profes methodology development. One new element, product-process dependency (PPD), was introduced to enhance the product quality focused process improvement process. In addition to working as a local project manager, the author was responsible for planning and co-ordinating the methodology development work package of the Profes project during 1998. During this period, the author was responsible for developing and managing the development of the first full version of the Profes methodology. More specifically, the author's main contribution and sphere of responsibilities were focused on the definition of Profes phases and steps. Furthermore, the author was working as a full time researcher in methodology development work during the whole course of the project. *Table 4* summarises the author's role concerning the related papers.

Table 4. Author's contribution to product quality focused research.

Year	Project	Author's contribution to the related papers
1997–1999	Profes	Papers V: Main author
		Papers VI: Profes methodology and design rationale.

1.3.4 Knowledge Management Enhancement

During the SPI method development and usage the author became more and more conscious of the fact that software development, and also SPI, involved capturing, using and managing an enormous amount of data. The Totem research project was initiated to study the knowledge processes associated with SPI (Totem 2001). To continue this research, the Finnish Academy funded the Knots-Q project, which was started in the year 2001 (Knots-Q 2002). The goal of this project was to develop knowledge-centred tools and methods for improving the quality of software production. Through these projects the first baseline for knowledge based SPI was encapsulated. *Table 5* summarises the author's role concerning the related paper.

Table 5. Author's contribution to KM enhancement.

Year	Project	Author's contribution to the related papers
1999–2000	Totem	
2000–2003	Knots-Q	Paper VII: Main author

1.4 Structure of the Thesis

This thesis is concerned with the development and use of SPI methods. The structure of this dissertation is presented in the following:

- Chapter 1 introduces the background of this research and outlines the research setting and the author's contribution to the research.
- Chapter 2 provides an overview to the various SPI research results relevant to this research. Related research is divided into SPI management research, software process best practices, measurement, product quality, and knowledge management. In SPI method development, many of these are attached to new or further enhanced SPI methods.
- Chapter 3 captures the SPI lessons learnt from literature using the results of industrial SPI case studies, SPI surveys and expert opinions. Using this information, the Critical Success Factor (CSF) criteria are developed for evaluating SPI methods.
- Chapter 4 evaluates the related research using the CSF criteria and showing the deficiencies and strengths of the various approaches.
- Chapter 5 presents and evaluates Pr^{2}imer, the integrated improvement management method. Pr^{2}imer is the first integrated SPI method to tie several SPI approaches into a single functional ensemble.
- Chapter 6 clarifies the role of measurement as an important part of the SPI initiative and proposes automating away repetitive tasks and some of the complexity associated with measurements and data management.
- Chapter 7 encapsulates and evaluates Profes, the product quality based process improvement method. Profes changes the SPI strategy by proposing the paradigm shift from process quality based improvement to product quality based improvement.
- Chapter 8 shows how to utilise knowledge management in SPI and discusses one practical and tested solution used for capturing and providing knowledge for a software engineering project.
- Chapter 9 sums up the research results in the light of the research questions and presents directions for further research.
- Chapter 10 recapitulates the original papers used in this thesis.

2. An Overview of Related Research

Various SPI methods and software process and quality standards have been researched actively since the late 1980's to support software development and software process improvement. For the purpose of this research, the related research relevant to executing SPI is structured into five categories: SPI management, process quality standards and appraisals, measurement, product quality, and knowledge management (KM). There are several potential ways of classifying approaches to improving software engineering activities. Kinnula (2001), for example, classifies available SPI related methods into two main broad classes, these being software process engineering process models and software process engineering infrastructure models. The categorisation applied in this research originates from the evolutionary path SPI method development has proceeded. The categorisation complements the taxonomy of Kinnula (2001) with measurement models, software process standards, product quality standards and knowledge management. *Table 6* presents the related research organised to categories originating from the SPI method development path.

Table 6. Related research.

SPI Management	Process Quality		Measurement	Product Quality	Knowledge Management
	Appraisals	Standards			
PDCA QIP Ideal ISO 15504-Part-7	SW-CMM Bootstrap ISO 15504-Part-2	ISO 9000-3 SWEBOOK	GQM SPC PSM BSC	ISO9126 IEEE Std 1061	Experience Factory

2.1 SPI Management Methods

In this section an overview is given of Deming's cycle (Deming 1986), Quality Improvement Paradigm (QIP) (Basili et al. 1994b), the IDEAL^{SM} model (McFeeley 1996), and ISO 15504 Part 7 (ISO 15504-7 1998). These methods propose an approach to managing an improvement initiative; furthermore, they are well known and commonly applied.

The basis for improvement management methods was established by Deming in 1986. Although he developed the 4-staged model for the needs of the manufacturing environment, the improvement principles have been applied in SPI. QIP, for example, represents a modified and fine-tuned model of the Deming's cycle in the context of software development. The IDEAL model divides improvement management activities into strategic and tactical levels. It has been developed to support SW-CMM based SPI. The aim of ISO 15504 Part 7 (ISO 15504-7 1998) is equivalent to that of IDEAL, except for the fact that the former has been designed to support ISO 15504 assessment based SPI. In the following, these models are briefly introduced.

2.1.1 Deming's cycle and TQC

The original Shewhart cycle (Shewhart 1931), later better known as the Deming or PDCA cycle (Deming 1986), was the first model to stress the importance of methodicalness and continuity in improvement actions. In addition to these aspects, the data involved in the planning and analysing phases was given a significant role. Ishikawa (1985) redefined Deming's cycle to six categories and named it as a Control Circle (*Figure 2*) within the Total Quality Control (TQC) improvement model. The model stresses the importance of the defined policy before establishing the improvement goals. Precise and purposefully expressed goals have to be based on the problems that the organisation needs to solve. The data supporting the control of the goal achievement needs to be clearly defined also.

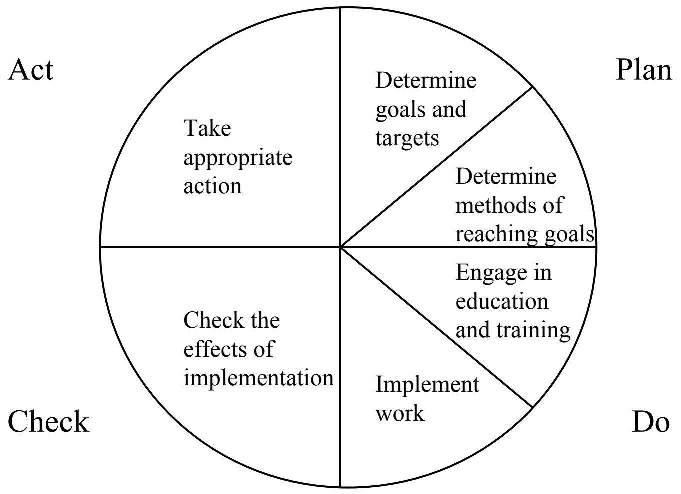
Act
Plan
Take appropriate action
Determine goals and targets
Determine methods of reaching goals
Engage in education and training
Implement work
Check the effects of implementation
Check
Do

Figure 2. Control circle (Ishikawa 1985).

The TQC approach emphasises that, to be successful, goal definition has to be accompanied by scientific and rational methods. It is even stated that otherwise nothing can be accomplished. While it is not explicitly fixed what is included in the scientific methods, the role of data and statistical data analysis is emphasised. The Cause and Effect Fishbone Diagram, also known as the Ishikawa Diagram, is presented as one method that can be used for collecting cause factors (called process), which may have an influence on implementing the desired quality characteristics. Here, the importance of the opinions of people who are familiar with the process in question is underlined. An open and frank atmosphere in the analysis sessions is promoted. The opinions need to be checked against the data available, so that the conclusions can be accepted by all and the first steps towards a standardised process or regulations can be taken. The standardised process is presented as a key success factor, even though the danger of over-standardisation and over-regulation is recognised. To avoid this pitfall and to highlight humanity and employee participation, the improvement statement is put forward as follows (Ishikawa 1985, p. 62): "Detailed standards and regulations are useless if they are established by headquarters staff and engineer-specialist who do not know or do not try to know the workplace and who ignore the wishes of the people who have to use them." Besides, TQC stresses that standards and regulations are imperfect and thus need to be reviewed and revisited regularly. In addition, education not only by giving lectures but also through actual work is highlighted. Regarding work standards, it is stated that if the standards are only distributed among workers, they may not read them, or they might not understand them correctly. Ishikawa (1985) further states that implementation is a straightforward action if TQC principles are followed, and therefore no special guidance for implementation is given. The implementation will merely be checked through the causes and the effects. First, it is studied whether all cause factors are under control. In practice this denotes checking if each of the processes conforms to the standard set. Secondly, the attainment of the wished effect on process or work is verified. In the checking step, the role of managers emerges, and checking is clearly seen as a function executed by managers, while workers receive feedback through the results gained. Taking an appropriate action as a concluding step means eliminating exceptions in wished effects. Appropriate actions are planned by studying the cause factors.

Despite the fact that Deming's circle and the TQC model were originally developed in the context of manufacturing industry, the improvement approach and philosophy have also been adapted to software engineering and SPI, e.g., in QIP, and therefore they can be regarded as relevant in the context of this research.

2.1.2 Quality Improvement Paradigm (QIP)

The Quality Improvement Paradigm (QIP) (Basili et al. 1994b, Basili & Caldiera 1995) can be seen as a fine-turned and more detailed model drawing upon Deming's cycle, and developed in the context of software engineering. QIP recognises three overall phases (planning, execution and evaluation) that comprise a total of six guiding steps for improvement actions (*Figure 3*). Compared to Deming's cycle, QIP introduces a new concept: experience packaging. What is learnt should be transferred to a form of experience package that could be utilised later (Basili et al. 1994b). QIP divides improvement activities into project and organisational levels. QIP is grounded on the idea that each project provides an opportunity for an organisation to learn about its processes, its products and related quality aspects, and to build and refine models for these objects.

The improvement approach incorporated into QIP is defined as an iterative process that repeatedly implements two feedback cycles, which are illustrated in

Figure 3. According to the QIP principles, the project level cycle incorporates feedback that is provided for the project during project execution. Furthermore, QIP stresses the use of data at project level for preventing and solving problems.

The project learning cycle provides two types of information for the corporate learning cycle. Firstly, project performance information is compared to existing project data and analysed regarding its concordance and ambiguity. Secondly, reusable and improved software assets that may be applicable to other projects are generalised and taken in use.

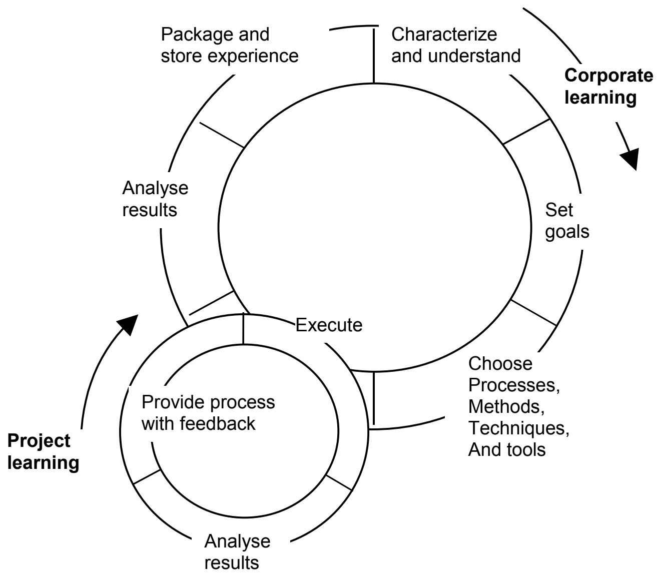
Package and store experience
Characterize and understand
Corporate learning
Analyse results
Set goals
Execute
Choose Processes, Methods, Techniques, And tools
Provide process with feedback
Project learning
Analyse results

Figure 3. The corporate and project cycles of QIP (Basili & Caldiera 1995).

The organisational learning cycle consists of the following steps: Characterise and understand, Set goals, Choose processes, Methods, Techniques and tools, Execute, Analyse results, and Package and store experience. The exact names of the steps vary slightly depending on the source.

The purpose of the “Characterise and understand” phase is to establish a baseline for any further actions by gathering knowledge of a project and its environment (organisation) regarding models and metrics that are in use. In the “Set goal” phase the goals are set for successful project performance and improvement. The baseline established in the previous phase is used for defining reasonable and quantifiable goals. The “Choose processes, methods, techniques and tools” phase describes models needed by a project to achieve the goals set earlier. The “Execute” phase consists of implementing the plans, collecting and validating the measurement data, and providing feedback to the project. In this phase, the operations are executed on a project level with the support of the organisation. After the project has been terminated the overall evaluation takes place in the “Analyse results” phase, in which project practices, problems, findings and recommendations are analysed. In the last phase “Package and store experiences” the structured knowledge, which may include models, metrics, lessons learnt and so on, is stored and made available to other projects. The QIP steps are performed repeatedly to achieve continuous improvement.

2.1.3 The IDEAL Model

While QIP proposes an open approach and ideology for managing improvement, IDEAL builds its improvement model on the process assessment results of SW-CMM, the Software Capability Maturity Model (Paulk et al. 1994). The IDEAL model developed by CMU/SEI is an improvement programme oriented model, which gives guidance on how to execute and manage an improvement programme (McFeeley 1996). The need to develop the IDEAL model arose from the application of SW-CMM: there was no guidance available on how to continue the work after the assessment. The model divides improvement activities into five phases: Initiating, Diagnosing, Establishing, Acting, and Leveraging. IDEAL recognises process improvement activities in two-dimensions: on strategic and tactical levels (*Figure 4*). When operating on the strategic level (Initiating phase), the processes that are of concern of senior management are the subjects of the improvement programme. On this level, SPI infrastructure is established, the improvement context defined, and the commitment to the improvement programme ensured. The improvement work is carried out on the tactical level (Diagnosing, Establishing and Acting phases) by line managers and practitioners. When entering into the Leveraging phase, the nature of improvement programme becomes strategic again. Then, the purpose is to review past activities and to make decisions for further actions.

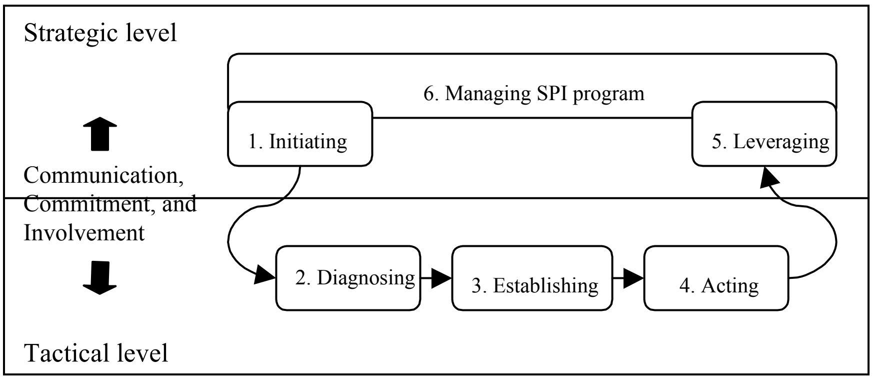

Figure 4. Two-dimensional view of the IDEAL model (McFeeley 1996).

The division between strategic and tactical level operations is identical to the QIP model, in which activities are divided into corporate and project levels. IDEAL can be seen as a Top-Down improvement approach where improvements are introduced to a software development project rather than developed on the basis of specific project needs.

2.1.4 ISO 15504 Part 7

In addition to ISO 15504, the process reference model (ISO 15504-2 1998) and assessment model (ISO 15504-5 1998), ISO has developed the ISO 15504 - Part 7: "Guide for use in process improvement" (ISO 15504-7 1998) to promote ISO 15504 assessment based SPI. ISO 15504 Part 7 is the counterpart to the SW-SW-CMM based IDEAL improvement model.

ISO 15504 Part 7 lists the SPI activities as follows:

1. Examine organisation's needs,
2. Initiate process improvement,
3. Prepare and conduct process assessment,
4. Analyse results and derive action plan,
5. Implement improvements,
6. Confirm improvements,
7. Sustain improvement gains, and
8. Monitor performance

In the ISO 15504 Part 7 model, the driving force of SPI can be found in the following statement: "software process improvement is based on process assessment results and process effectiveness measures" (ISO 15504-7 1998, p. 2). Although the starting point of improvement based on assessment is visible also in the IDEAL model, it is more clearly emphasised in ISO 15504 Part 7 (1998). ISO 15504 is developed by the Spice project thus it is often referred as Spice method as well.

2.2 Software Process Best Practices

This chapter introduces approaches that aim to improve software development using the knowledge embodied in the form of good software development practices. Based on the way this knowledge is used, two perspectives may be distinguished: the use of software process standards and the use of process assessment methods. Process standards are used as examples of the best practices that are adapted as is or reworked to fit the needs of the subject company. Software process assessment methods evaluate the maturity of the software development processes in the company, comparing them to the reference process model the method is based on. Nonetheless, the exploitation of the ideas of good software development process is similar in both of these approaches.

2.2.1 Assessment Based Approaches

The core idea of assessment-based SPI is that software practices should be organised according to the reference process model. Humphrey presents the principles of assessment-based SPI in his work "Managing the Software Process" (Humphrey 1989).

Assessment-based SPI approaches concentrate on assessing existing software development processes and comparing them with the specific reference process model the particular assessment method is based on. The used technique to conduct an assessment is to interview personnel using structured questionnaires. The assessment results are an indication of how well existing processes fulfil the requirements of the method.

In the sections below, the widely used (ref. e.g. Debou et al. 1999, Goldenson & Herbsleb 1995, Haley 1996, Hollenbach et al. 1997, McGuinnes 1996, Lanzerstorfer & Scherzer 1999) SEI assessment method SW-CMM (Paulk et al. 1993, Paulk et al. 1994) and the newcomer CMMI (2000) are summarized. In addition, the Bootstrap method, the European counterpart to the above-mentioned methods, will be shortly presented. Lastly, ISO 15504, or Spice (Emam et al. 1998), i.e., the output of an international research effort, is introduced.

2.2.1.1 SEI Capability Maturity Models

SW-CMM, the Capability Maturity Model for Software, is the oldest, the best known and the most applied assessment method (name refined from CMM) provides an assessment driven approach to SPI. It was published by SEI (Paulk et al. 1994, Paulk et al. 1993) and it has been widely used ever since. It was originally developed for assessing the capability of DoD contractors (Humphrey & Sweet 1987), but it soon became a reference guide for subcontractor SPI and shortly after that a guideline for any organisation seeking to improve its software processes. SW-CMM organises software development practices (called Key Process Areas) to five maturity levels (*Figure 5*). The organisation of processes is not justified, for which the method has received criticism. Each maturity level builds on its predecessors, which is why reaching a higher maturity level requires that all practices must be fulfilled at lower levels. For this reason, the organisation must progressively implement all the practices one by one at each level from levels 2 to 5. When SW-CMM is applied to the letter, it is likely to restrict the flexibility organisations might need to improve their software development to best results.

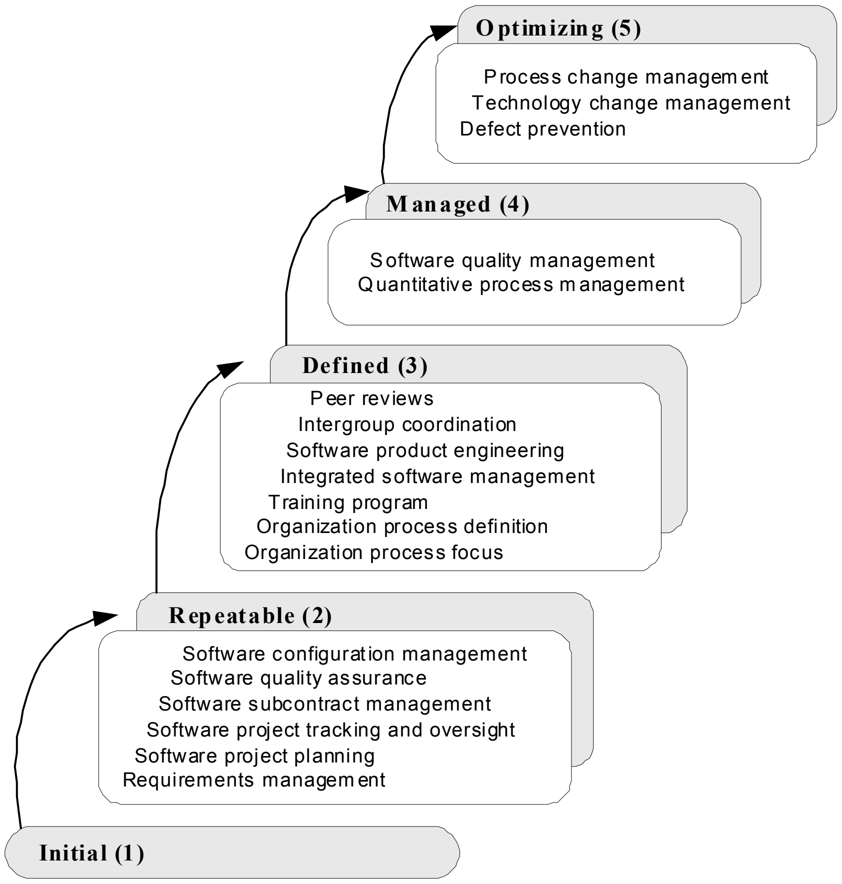
Optimizing (5)
Process change management
Technology change management
Defect prevention
Managed (4)
Software quality management
Quantitative process management
Defined (3)
Peer reviews
Intergroup coordination
Software product engineering
Integrated software management
Training program
Organization process definition
Organization process focus
Repeatable (2)
Software configuration management
Software quality assurance
Software subcontract management
Software project tracking and oversight
Software project planning
Requirements management
Initial (1)

Figure 5. SW-CMM Key Process Areas by maturity level (Paulk et al. 1993).

Despite its broad usage, the further enhancements of SW-CMM were stopped and development efforts were directed from 1998 onwards to the Capability Maturity Model-Integrated (CMMI 2000). SEI thus started to integrate existing capability models and to develop a new integrated model. CMMI integrates SW-CMM and SE-CMM (1995), of which the latter was developed for assessing systems engineering processes. A further CMM Integration project goal has been to include features from other models such as EIA/IS-731 (1998) and IPD-CMM v0.9a (1997). According to (CMMI 1999) the new integrated model aims to be compliant with the ISO 15504. CMMI has adjusted the assessment approach according to Bootstrap (Kuvaja et al. 1994) and ISO 15504 (ISO 15504-5 1998). CMMI recognises two approaches to assessment and improvement: a traditional staged model and a continuous model allowing one to work with only selected process areas. CMMI has defined the A, B and C types of assessment: Class A is the most formal appraisal and it is conducted by an official lead assessor; this class is often called the SCAMPI (Standard CMMI Assessment Method for Process Improvement) assessment. SCAMBI Class B is a less formal appraisal for investigating process capabilities without producing ratings of process capability. Class C assessment, again, is a quick look over the risk areas of processes done by the organisation itself (SCAMPI 2001).

A possible future development of the SW-CMM model was, again, under discussion at the end of 2002. A press release (Watzman & Perdue 2002) announced the continuation of SW-CMM, and later this information was abrogated by SEI.

2.2.1.2 Bootstrap

The European counterpart to SW-CMM is the Bootstrap assessment method developed using several standards as the ISO 9000 series, ESA PSS-05-0 (1994), the European Space Agency standard, the DoD standard DoD-STD-2167A, and SW-CMM (Kuvaja et al. 1994). Compared to SW-CMM, the Bootstrap method widened the scope of assessment activities. While SW-CMM focuses on project level processes, Bootstrap also assesses the software development organisation and its processes. At the organisational level, the purpose is to clarify what kind of assets the subject organisation is capable of providing for software development projects (e.g. quality manuals and instructions). In addition to assessing organisational processes such as “human resource management”, Bootstrap applies a fine-turned evaluation approach: instead of just “yes” and “no” the evaluation answers may vary between “fully”, “largely”, “partially”, “not” and “not applicable (NA)” (BootCheck 1997). Another assessment reform Bootstrap carried out was the possibility to select a set of processes for assessment.

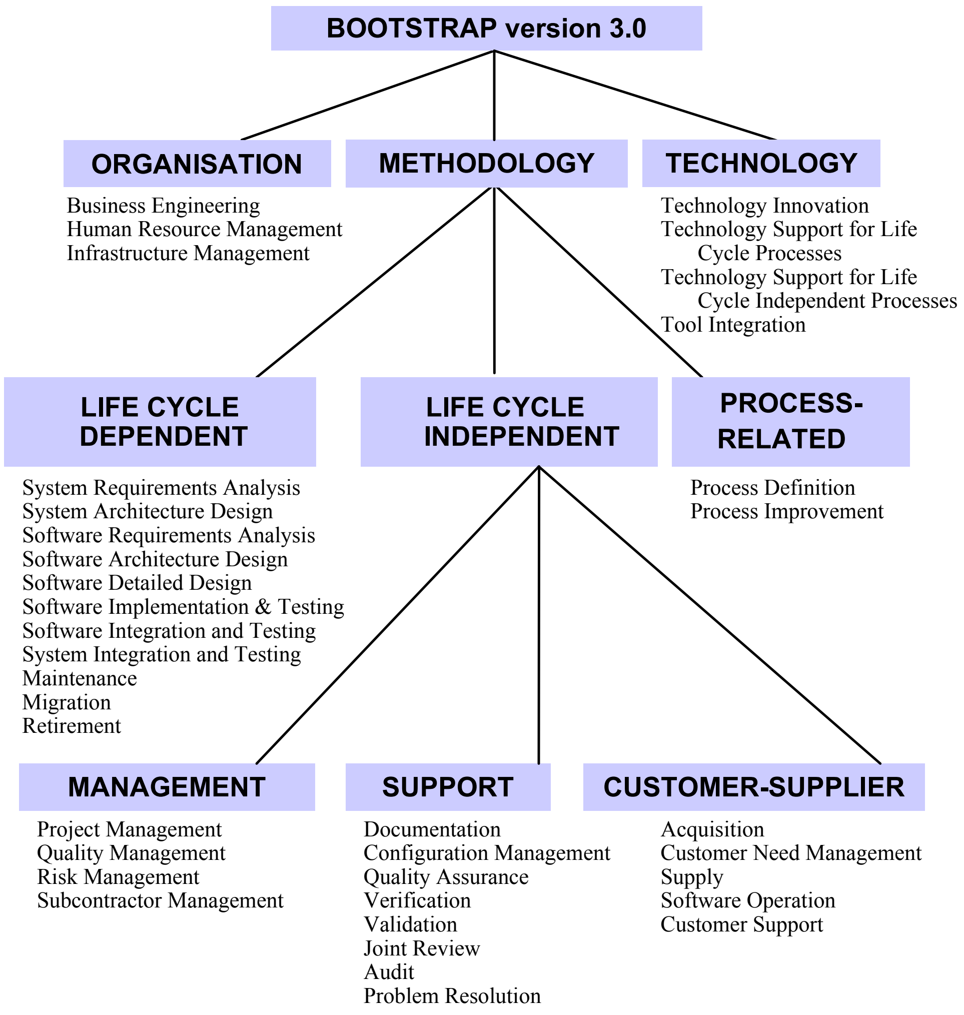

Figure 6. The Bootstrap version 3.0 process architecture (Bicego et al. 1998).

Bootstrap decouples the process model and the capability model. In Bootstrap, the improvement philosophy states that improvement should be driven by organisational needs, whereas the process model provides an outline for the improvement of individual processes. Compared to SW-CMM, the organisation maturity level is replaced by process capability profiles showing the capability level of each process.

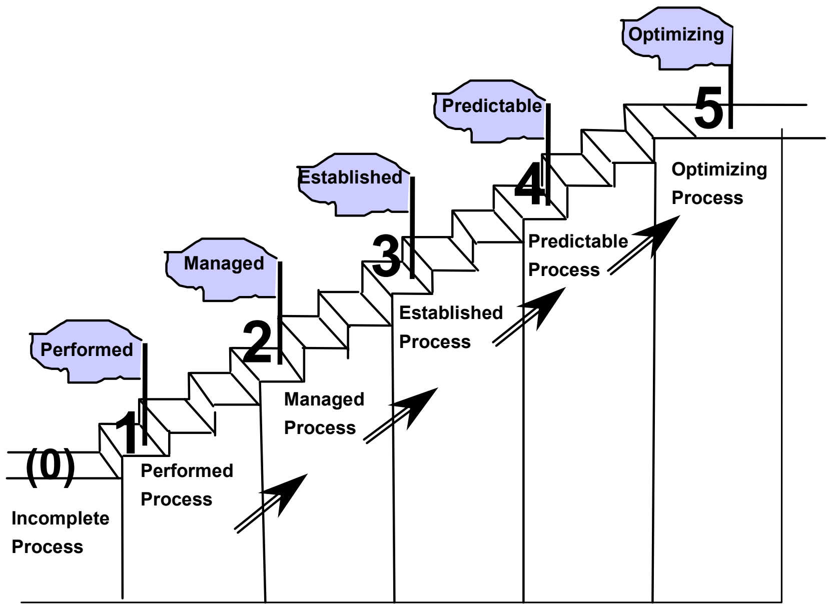

Figure 7. Bootstrap capability levels.

The capability of processes is rated from 1 to 5, but it is further defined by quarters (*Figure 7*). For example, the capability of "risk management" may be 1.75 or that of "quality assurance" 2.5.

Until Spring 2003 the Bootstrap method was only available under a licence, which may have prevented a large-scale use of it.

2.2.1.3 ISO 15504 (Spice)

To answer the harmonising need for assessment, the International Standardisation Organisation (ISO) established the Spice project to carry out the standardisation process (Drouin 1999). ISO had previously published software life cycle processes in the ISO 12207 standard (1995/2002) that formed a basis for the reference process model for ISO 15504 Part 2 (1998). The development of ISO 15504 brought experts together from all over the world. The goal of the project was to produce an assessment method for organisations of different sizes, application domains, and management styles that may have different improvement priorities. The ISO 15504 method includes a process model with six capability levels, and a set of reference processes aligned to the ISO 12207 definition. The capability levels of ISO 15504 are applied to each individual process. No predefined sequence is demanded which means that priorities are not fixed to improve certain processes. Priority definition is based on each organisation's requirements and business goals.

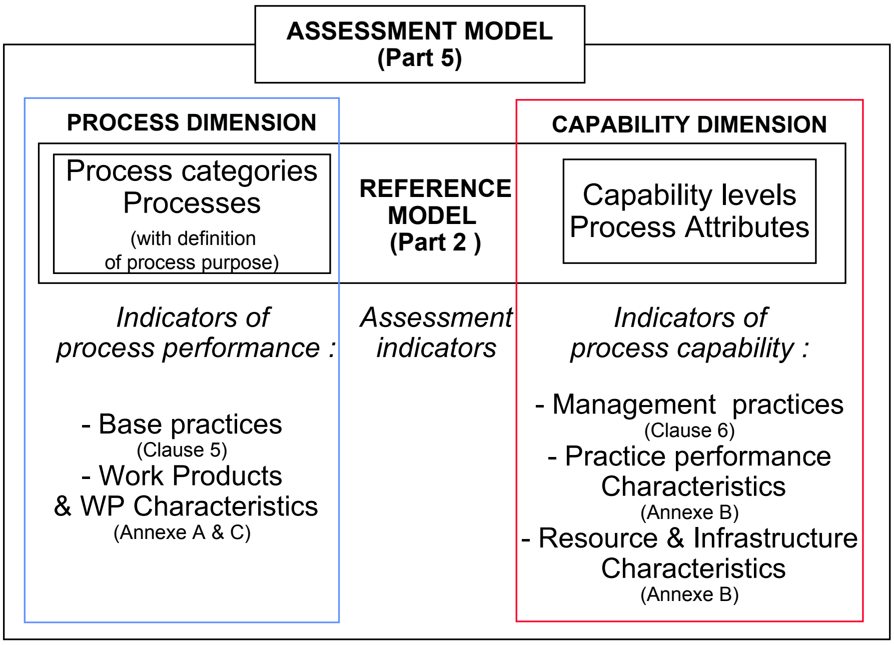

Figure 8. Relationship between the ISO 15504 reference model and the assessment model (ISO 15504 Part 5, 1998).

Process performance, in other words the achievement of the process purpose, is evaluated using the base practices associated to each process. Process capability is rated by assessing the demonstrated achievement of sets of management practices associated to the different capability levels (0–5). The assessment model is illustrated in *Figure 8*.

2.2.2 Software Process Standards

Software process standards document a standardised definition of software development practices. They are often produced by software acquirers such as the Department of Defence (DoD) in the USA or the European Space Agency (ESA) for the use of subcontractors. This being the case, the use of standards may be mandatory or at least recommended. Standards define also vocabulary and terminology, which should help the communication and clarify the expectations between stakeholders. All software developing organisations may use the available standards as reference process models for improving their processes. An example of a widely applied standard in software development is ISO 9000-3 (1997), either used for establishing ISO conformant software development processes or for acquiring the official ISO 9000 certificate.

Since from the SPI work point of view all process standards share the same principles, only a brief overview of the well-known ISO 9000 series and the ongoing international SWEBOK effort is given in the following. Due to the nature of the ISO 15504, it is presented in Chapter 2.2.1, in which assessment based SPI approaches are introduced.

2.2.2.1 ISO 9000 series

Several international standards are used largely as reference models when building or improving software quality and development practices. ISO 9001 (2000) provides a model for quality assurance in design, development, production, installation, and servicing. It contains basic requirements for building and maintaining a quality system. ISO 9000-3 (1997) provides guidelines for the application of ISO 9001 in software development. The advantage of these models is that they are not too complex, which has made them useful and popular in industry in defining software development practices. Unfortunately, due to the static nature of standards the drawback is that they do not provide any guidance on improvement actions.

2.2.2.2 SWEBOK

The ongoing international SWEBOK project (http://www.swebok.org/) aims to "provide a consensually validated characterisation of the bounds of the software engineering discipline and to provide a topical access to the Body of Knowledge supporting that discipline" (SWEBOK 2001, p. i). The project is promoted by the IEEE Computer Society and the ACM. The aim of this international project is to provide a consistent view of software engineering for private and public sectors. SWEBOK defines nine knowledge areas for software engineering:
– Software requirements
– Software construction

- Software testing
- Software maintenance
- Software configuration management
- Software engineering management
- Software engineering process
- Software engineering tools and methods
- Software quality.

All these areas and their sub-areas are described in detail. The project is currently in a trial phase and the final version of SWEBOK should be available after the end of the year 2003, when the third phase of the project called the "Iron Man Phase" is scheduled to be finalised.

2.3 Measurement

Measurement is a mean to acquire quantitative information of software development processes and products for the purpose of managing them. Measurement can be used to define the status of processes or product quality, to analyse the effects of changes, or to follow-up the progression of improvement actions. Here, four different measurement methods are introduced: Goal/Question/Metric (Basili & Weiss 1984, Basili et al. 1994a, van Solingen & Berghout 1999), statistical process control (Florac & Carleton 1999, Florac et al. 1997), practical software measurement (PSM) (PSM 2000) and, lastly, the balanced score card (BSC) (Kaplan & Norton 1996, Olve et al. 1999). These are all well-known measurement methods applied in software process management and improvement and thus shortly described here.

2.3.1 The Goal/Question/Metric Method (GQM)

During the 1990's the Goal/Question/Metric (GQM) measurement method introduced by Basili & Weiss (1984) matured from state-of-the-art (Basili et al. 1994a) to state-of-the-practice (van Solingen & Berghout 1999). The GQM method built on the QIP paradigm (Basili et al. 1994b, Basili & Caldiera 1995) aims to provide information needed for understanding, guiding, and changing the software processes of a software development project.

The GQM method represents a systematic top-down approach to defining and collecting measurements and, on the other hand, a bottom-up approach when analysing data against stated measurement goals. One of the method's main aims is to establish a visible link from measurement goals to the data collected. The underlying idea is to avoid the high risk of wasting resources when measurement data is collected without an idea of its usage. GQM adapts and integrates organisational objectives into measurement goals, and refines them into measurable attributes on a step-by-step basis; therefore, GQM helps to identify the exact metrics necessary for meeting case-specific objectives.

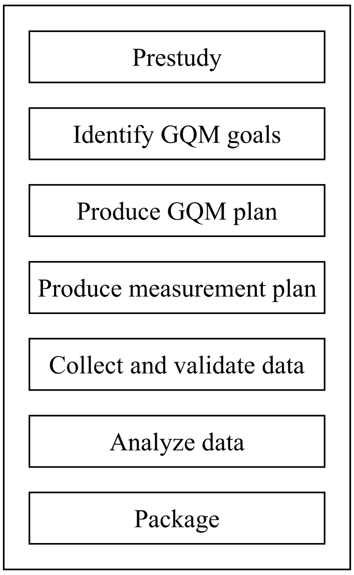

Figure 9. The activities of a GQM measurement programme (Gresse et al. 1995).

Unlike assessment-based approaches, GQM is not based on any software best practice model. First and foremost, GQM is a method for defining measurements according to measurement goals and therefore, from the SPI point of view, GQM users are not directly supported to identify what to improve or how to improve the performance. However, GQM can support the fine-tuning of improvement initiatives in the course of process improvement, it provides methodological support for defining metrics used for monitoring the results of process changes during and after an improvement initiative.

The GQM method has been widely applied in software engineering and it has become the de-facto standard in the field of measurement.

2.3.2 Statistical Process Control (SPC)

Meaningful SPC may take place when functional measurement practices and environment already exist. Florac & Carleton (1999) suggest some other measurement methods to be used for establishing measurement activities before applying SPC. SPC focuses mainly on analysing process performance using the control chart principles, but also suggests a six-step strategy to be applied in the measurement programme. These steps are introduced in *Figure 10*.

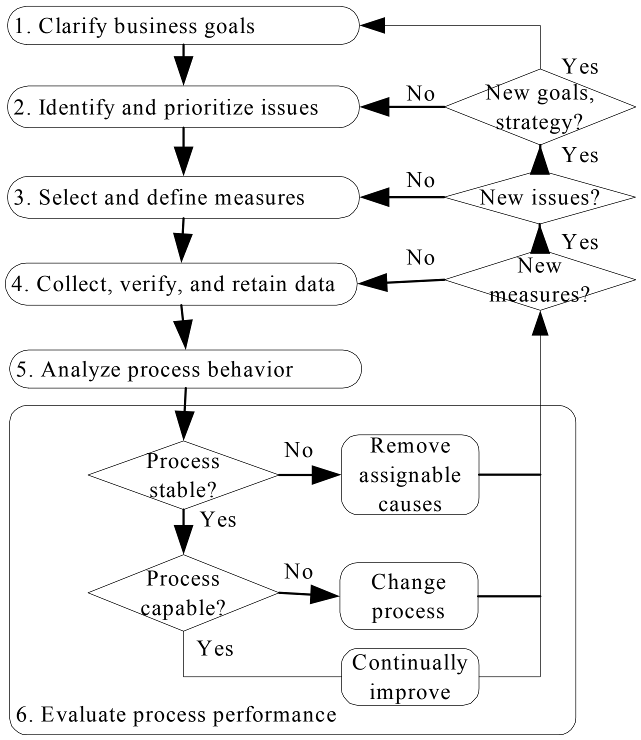
1. Clarify business goals
2. Identify and prioritize issues
3. Select and define measures
4. Collect, verify, and retain data
5. Analyze process behavior
New goals, strategy?
New issues?
New measures?
Yes
No
Process stable?
Process capable?
Remove assignable causes
Change process
Continually improve
6. Evaluate process performance

Figure 10. SPC measurement process (Florac & Carleton 1999).

The main emphasis has been placed on clarifying the statistical means in step 5. Using any statistical analysis requires a larger sample; to conduct any reasonable analysis at least four or five similar kinds of projects will have to be involved.

However, SPC provides help for measurement analysis and underlines the importance of linking measures with the business goals of an organisation.

2.3.3 Practical Software Measurement (PSM)

The development of the Practical Software Measurement (PSM) method was initiated by DoD (Sanders 1997). The purpose of this issue-based measurement method is to guide project managers in selecting, collecting, defining, analysing and reporting the specific software issues and objectives of each program. To be successful, projects have to be able to manage several issues: e.g. objectives, risks, lack of information, and problems (PSM 2000, part 1). The focus of PSM is primarily on individual project level measurement, while the measurement should be guided by the concerns, objectives and the context of the project.

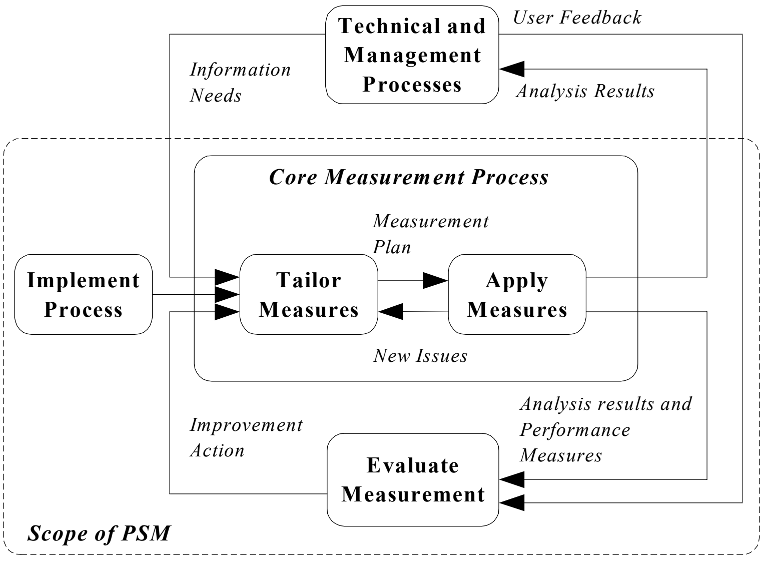

Figure 11. PSM measurement process (PSM 2000, Part 1).

PSM divides the measurement program into five main phases, each of them including several sub-activities. The PSM core measurement process consists of two phases: Tailor Measures and Apply Measures (*Figure 11*). In the Tailor Measures phase, the project prioritises project issues, selects and specifies measurements and integrates them to the project life cycle. In the Apply

Measures phase, measurements are collected, analysed, and based on analysis recommendations are given. PSM emphasises the fact that the measurement analysis should be done by persons familiar with the project context. In the Evaluate Measurement phase, the measurement program itself is assessed and improvements are proposed. The Technical and Management Processes phase is external to PSM because it describes the technical and management processes of each software development project. These can be carried out by an external unit if software development is subcontracted. The maturity of technical and management activities may have an influence on which measurements it is meaningful to collect. In the Implement Process phase, the environment for carrying out measurement activities is ensured; the cultural and organisational changes needed are addressed, resources provided, and practical support for managers and teams is given.

2.3.4 Balanced Score Card (BSC)

The balanced scorecard (BSC) is an organisational management system with a strong measurement emphasis. This top-down approach aggregates four different types of stakeholders under one management system. The development of BSC started in 1990 from the notification that financial measures solely do not provide enough information for properly managing organisations operating in a complex environment. In addition to financial measurement, three other views were included: customer, internal business process, and learning and growth (*Figure 12*). The goal of BSC is to make the organisation strategy specific and actionable, to engage everyone in the organisation in target setting, and to provide feedback and learning. A successful use of BSC requires that the organisation strategy be translated to the language of the various stakeholders. BSC recognises the following measurement steps: define metrics, collect data, analyse data, and decide on changes. (Kaplan & Norton 1996).

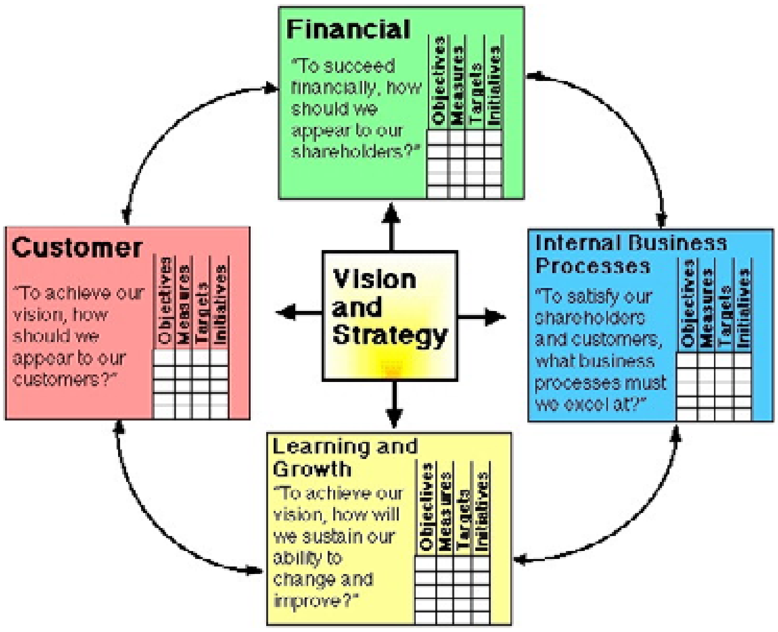
Financial
"To succeed financially, how should we appear to our shareholders?"
Objectives
Measures
Targets
Initiatives
Customer
"To achieve our vision, how should we appear to our customers?"
Objectives
Measures
Targets
Initiatives
Vision and Strategy
Internal Business Processes
"To satisfy our shareholders and customers, what business processes must we excel at?"
Objectives
Measures
Targets
Initiatives
Learning and Growth
"To achieve our vision, how will we sustain our ability to change and improve?"
Objectives
Measures
Targets
Initiatives

Figure 12. BSC structure (Kaplan & Norton 1996).

Olve and others (1999) describe an 11-step procedure for building BSC. These steps start on defining the context of the organisation, establishing the vision and perspectives of the organisation, breaking down the vision to strategic goals with critical success factors, and developing a balanced, top level BSC with measures, causes and effects. BSC is further broken down according to organisational units, formulating lower level goals and an action plan for guiding the implementation of BSC. Providing a remarkably broad view on management and measurement, BSC has the capability to link SPI with an organisational context.

2.4 Product Quality

The aim of SPI actions is often to raise the quality of software. Despite this, there are not that many standards or methods focusing on the quality of software products. The oldest and best-known method is the recently updated ISO9126-1 (2001) standard for software product quality. In addition, IEEE has published the standard for software quality metrics methodology (IEEE Std 1061-1998). Both of these standards pinpoint the definition of software quality requirements and identification of metrics needed for following and assessing the satisfaction of them. There have been research attempts towards defining more detailed software product quality methodologies and assessment frameworks at European level such as the Esprit 4 project SPACE-UFO (1998) and Esprit 2 project SCOPE (1993) but, unfortunately, the results of these projects have not been widely adopted, if at all, by industry.

In the following, the ISO9126 and IEEE 1061 standards are recapitulated.

2.4.1 ISO 9126

The ISO 9126-1 (2001) standard divides product quality into six characteristics: functionality, reliability, usability, efficiency, maintainability and portability. These characteristics are further broken down into several sub-characteristics. *Figure 13* presents an overview of the ISO 9126 standard. While the standard ISO 9126 has received some criticism (Mellor 1992), its advantage can be found in the fact that it suggests a common vocabulary to use.

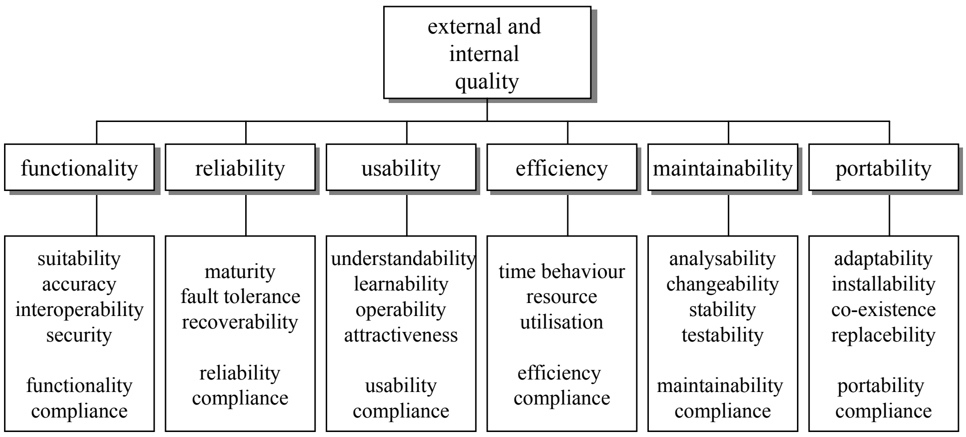
external and internal quality
functionality
reliability
usability
efficiency
maintainability
portability
suitability accuracy interoperability security functionality compliance
maturity fault tolerance recoverability reliability compliance
understandability learnability operability attractiveness usability compliance
time behaviour resource utilisation efficiency compliance
analysability changeability stability testability maintainability compliance
adaptability installability co-existence replacebility portability compliance

Figure 13. The ISO 9126 model (2001).

In addition to the quality model presented above, the ISO 9126 standard also recommends a three-stage evaluation process model by which software product quality assessment may be performed. The advocated assessment process involves the definition of quality requirements for a product, the selection of appropriate metrics and the collection of measurements from selected product parts (ISO 14598-1 1999).

2.4.2 IEEE Std 1061

The IEEE standard 1061 (IEEE Std 1061, 1998) guides the setting of quality requirements and software metrics, while, unlike ISO 9126 (ISO 9126-2 2003, ISO 9126-3 2003), it does not describe any specific metrics. Furthermore, it gives instructions on how to implement, analyse and validate software quality metrics. For an informative framework, the standard proposes the GQM paradigm (Basili et al. 1994a) or PSM (PSM 2000) to be considered when establishing a metric framework for an organisation.

2.5 Knowledge Management

Knowledge Management (KM), SPI and software engineering research along with their relationship have been of the interest of many researchers and practitioners. As is known, software engineering itself is extremely knowledge intensive and creative features-bearing discipline (Ruhe 2001, Rus & Lindvall 2002). Furthermore, given that only part of the knowledge is documented, creating, capturing and sharing knowledge effectively is a major challenge for organisations.

In the following, KM research and terminology is recapitulated. Furthermore, Experience Factory (EF), which uses KM principles, is introduced. The EF concept developed in the field of software engineering put forward an organisational structure that supports knowledge creation and usage.

2.5.1 KM Research

The organisational knowledge creation theory of Nonaka & Takeuchi (1995) assumes that knowledge is created through social interaction between tacit and explicit knowledge. It is rather difficult to formalise or communicate tacit knowledge, as it comprises personal experience, ability, and beliefs. Tacit knowledge is associated with personal intention and commitment. A simple example of the use of tacit knowledge would be the skill to ride a bike. Explicit knowledge, again, refers to a formal form of knowledge that is transformable using systematic language.

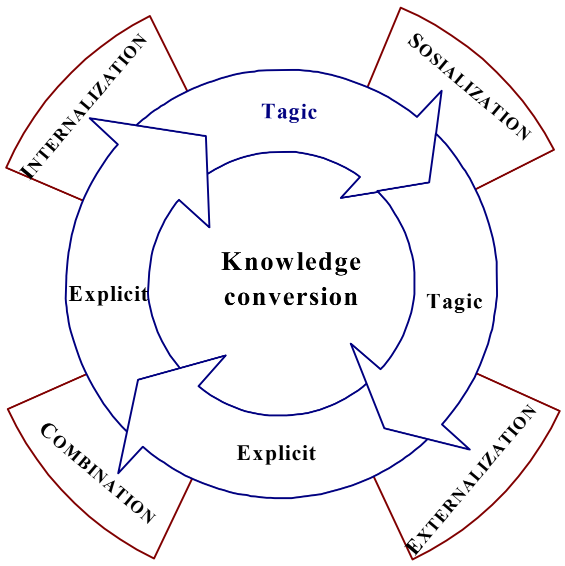

Figure 14. Four modes of knowledge conversion.

In Socialization, experience is shared between individuals using observation and imitation, no language is used. For example, by observing we learn the unwritten norms and values that a home, organisation, or country may have. Acquiring tacit knowledge is a gradual knowledge creation process. When externalisation takes place, tacit knowledge is verbalised and transformed to explicit knowledge. The purpose is to formalise knowledge to reusable and transformable forms such as memos, documents, models, concepts, formulae etc. Combination covers activities aiming to create new knowledge by utilising explicit knowledge. Combination occurs when manipulating the knowledge system, e.g., by adding, sorting, or categorising in order to create new understanding based on existing explicit knowledge. In Internalisation, explicit knowledge transmutes again to personal tacit knowledge. By reading, or listening, explicit knowledge may become tacit knowledge. This way we may benefit from the experiences of others and avoid doing the same mistakes as others, and also find ways to improve our actions.

KM within software engineering and SPI is addressed from several perspectives, e.g., Kucza and others (2001) have studied the interface between the software reuse process and KM. Dingsøyr (2002) has investigated how intranet-based KM tools are used in organisations to support software development. Kneuper (2002) has studied the knowledge needs of software developers and the means of providing the needed knowledge for them. Birk et al. (2002) emphasise the post-mortem analysis of software development projects as an efficient way to initiate KM in small- or medium size projects. Pourkomeylian (2002) has researched the challenges of SPI from the KM point of view. Before KM became well known as a research subject, Basili and others (1994b) developed the Experience Factory concept embodying the prominent KM terminology defined by others (Davenport & Prusak 1998, Nonaka & Takeuchi 1995).

2.5.2 Experience Factory (EF)

Davenport & Prusak (1998) define that Experience refers to the past events and it provides a historical perspective for predicting forthcoming events. Furthermore, Knowledge arises from experience; it is attached to values, context information, and individual comprehension and insights. Nonaka & Takeuchi (1995) divide knowledge into two classes, tacit and explicit knowledge.

In pursuance of KM principles, Basili and others (1994b) claim that an efficient and systematic reuse of experience necessitates an organisational structure that supports it. They call this the organisational structure Experience Factory (EF). The EF concept builds on understanding that product and SPI assume a continuous accumulation of experience, in other words combination of explicit knowledge. EF is illustrated in *Figure 15*.

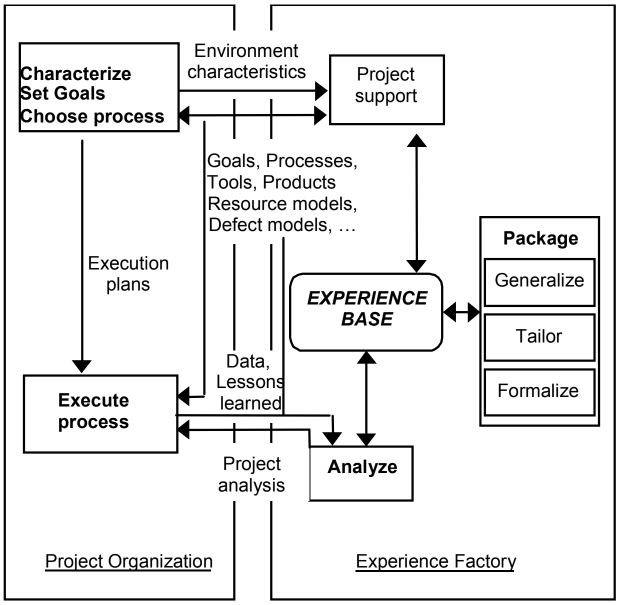

Figure 15. Experience Factory (Basili et al. 1994b).

The EF concept consists of two separate organisations: Project and Experience Factory organisation. The latter is further separated to two parts: analysis and support organisation. Software project exploits the models and experience provided by the analysis organisation. The project organisation concentrates on software development obeying models that support reuse, but is expected to provide project related information such as project and process characteristics, development data, cost and schedule information, quality records, and feedback to the analysis organisation. The analysis organisation provides on demand artefacts analysed to suit the project. The support organisation facilitates communication by taking care of interactions between developers and analysts. In addition, it is responsible for experience management from the information technology point of view, i.e., taking care of packaging, storing and retrieving project experiences (Basili et al. 1994b).

The basic principle in developing EF is the notion that experience sharing is not a matter-of-course within organisations and projects, hence it has to be organised and managed systematically. The knowledge repository, where the packaged, organised and reusable experience packages are stored and retrieved from is called Experience Base (EB). Experience packages may include information of products, processes, practices, methods, lessons learnt, techniques, and tools etc. (Basili et al. 1994b).

EF puts forward a framework for managing experiences according to the QIP principles and offers facilities for utilising packaged software development knowledge as an input to SPI.

2.6 Discussion

SPI related research and interest has been very active both in academia and in industry. As a result, various assessment and measurement based SPI methods have been introduced in addition to the SPI management methods, quality standards, and solutions for managing SPI knowledge. The proposed methods have been applied to SPI by various practitioners with manifold results. There are quite a few success stories (Herbsleb et al. 1994, Dyne 1998, Fitzgerald and O'Kane 1999) as well as recognised method limitations (Card 1991, Lehman 1995) and failures in achieving goals (Goldenson & Herbsleb 1995, El Emam and Smith 1999, Conradi & Fuggetta 2002, Kautz & Nielsen 2000).

As noted by many researchers (Quinn 1996, Conradi & Fuggetta 2002, Humphrey 1989), SPI is a complex activity requiring diverse expertise. Individual SPI approaches are often recommended and thus also selected by industry as a mean to solve a quality or other software development related problem. The different methods may even have been seen as competitors to each other. SPI methods are often applied separately, and furthermore, there seems to be a lack of understanding as to how to integrate them to achieve better results.

3. Development of the SPI Method Evaluation Criteria

In this Chapter, the findings of the literature review of SPI success factors are presented and analysed. The review sources include 11 SPI case study reports covering 35 SPI cases and the results of three large SPI surveys, which are augmented with three expert opinions based on long-term involvement in SPI. The ultimate purpose of this review is to develop *Critical Success Factor (CSF) criteria* for evaluating SPI methods.

Although the concept of CSF was first introduced by information system development as early as the late 70's, it has not yet been widely used by software development (Fitzgerald & O'Kane 1999). The application of CSF criteria is now introduced for the first time in the area of process improvement. Many researchers and practitioners have reported experiences of SPI arguing for various reasons for success. Here, these statements are utilised as building blocks in defining the CSF criteria. Later, the CSF criteria are used to evaluate related research and SPI methods developed during the four engineering research process cycles.

In the following sections, first the background of the CSF criteria development is clarified including related research. Second, the unique SPI cases, survey results and expert opinions are shortly presented in an author-centric way, as guided by (Webster & Watson 2002). Appendix B presents success factor statements as captured from literature. Using the findings of the review, based on success factor statement evaluation and synthesis, SPI success factor statements are compiled to seven categories: improvement management, commitment, culture, and four general SPI engineering activity classes (plan, do, check, action). In some of these categories, there were competing statements, the most often highlighted one of which was then selected and formulated in a form of CSF proposition. The CSF criteria consist of critical propositions, the fulfilling of which is likely to lead to improvement success. The definition of improvement success is case dependent; it may mean improving the quality of software in terms of defects, achieving a specific quality certificate, or decreasing the time to market, for example.

3.1 Background

The fundamental assumption is that the issues pinpointed as SPI lessons learnt are explanatory factors for the results gained and the explicitly stated success factors are elements facilitating improvement, at least in the individual cases they were extracted from. When a lesson or a factor is raised again and again, it becomes a general canon. Thus, SPI literature is analysed to extract common premises and components for successful improvement actions. These factors are further studied to specify how they could be supported by an SPI method. As a result, the requirements for an SPI method are generated.

3.1.1 Related Research

Objective evaluation and comparison of SPI frameworks and supportive methods is difficult: they differ in scope, in detail, and also in the basic philosophy they apply. In spite of this, literature reveals a few ways to evaluate SPI frameworks and methods. Halvorsen & Conradi (2001) developed the Characteristics comparison method and came up with the following four SPI method comparison classes:

1. *The Characteristics comparison method* is based on a list of characteristics, as objective, measurable and comparable as possible, such as popularity, assessment, organisation size, quality perspective, certification. To this class Halvorsen & Conradi (2001) developed a total of 25 characteristics and divided them in five classes (general, process, organisation, quality, and result).

2. *The Framework mapping comparison method* concentrates on creating a map from the kernel statements or concepts of one framework to those in other frameworks. The mapping is useful, for example, for organisations wishing to apply several methods and desiring to eliminate redundant work. Framework mapping is a more detailed analysis than Characteristics comparison method as comparison is done at more specific levels. An example of this kind of mapping is to link the ISO 9001 Clause 4.6 "Purchasing" with the SW-CMM Key Process Area "Software Subcontract Management" (Paulk 1995).

3. *The Bilateral comparison method* compares two frameworks and summarises the findings in textual descriptive format: “ISO 9001 requires an organisation to be able to identify and trace a product through all stages of production, delivery, and installation. The SW-CMM covers this clause primarily at level 2 in the context of configuration management, but states the need for consistency and traceability between software work products at level 3” (Paulk 1995).

4. *The Needs mapping comparison method* does not compare methods to each other but uses needs and goals of an organisation in a method selection, as an example an organisation may have to adjust processes according to ISO 9000 standard to get the certificate and thus be able to win contracts.

The first three of the methods presented above are developed for comparing the characteristics of SPI methods, for creating maps between their statements, and for analysing the software engineering statements incorporated in the methods. Although these comparison methods provide useful information of SPI methods, they are, unfortunately, limited in analysing them from the viewpoints of effectiveness and success.

Dybå (2000) has developed an instrument for measuring the key factors of success in SPI. The instrument contains six key factors for SPI success and 37 related indicators that can be used for guiding or evaluating SPI actions. The six key factors developed using literature reviews, expert opinions and company studies are:

1. Business orientation
2. Leadership involvement
3. Employee participation
4. Concern for measurement
5. Exploitation of existing knowledge
6. Exploration of new knowledge

The instrument uses scales from 1 to 5 to evaluate a single indicator. For example, the leadership involvement key factor comprises five indicators, e.g., “Management accepts responsibility for SPI” and “SPI issues are often discussed in top management meetings” (Dybå 2000, p. 372). The respondent rates these indicators using a given scale (strongly disagree – disagree - neither agree nor disagree - agree - strongly agree). Dybå’s measurement instrument is designed for measuring to what extent the conducted SPI actions support extracted SPI key factors, but not for evaluating the characteristics of SPI methods.

3.1.2 Development Process

The purpose of CSF criteria development has been to build a set of criteria for evaluating how SPI methods support critical SPI success factors in establishing and in the course of an SPI initiative. The underlying idea is that if general SPI CSFs can be extracted, they should be valued by an SPI method as well. CSFs stand for issues that are generally important for successful SPI and therefore they should be supported by SPI methods. Furthermore, the hypothesis is that the more CSFs are supported the lower the risk of failure and the higher the possibilities for SPI success.

Since the related research review did not reveal any suitable means for evaluating SPI methods, it was decided to develop a new, generally applicable set of CSF criteria. The criteria is developed on the basis of the related literature review. The scope of the review covers software quality related conferences and magazines. These are expected to provide the best sources for industrial case study reports of SPI. In addition, the book by Messnarz & Tully (1999) provides versatile set of industrial SPI cases for evaluation. The search was completed using a literature review completion condition as proposed in (Webster & Watson 2002). At the point of review completion found new articles did not provide any further success factor statements to be considered.

Based on the review of SPI lessons learnt and explicitly stated success factors, the unique success factor statements are compacted to 15 CSF propositions linked to seven classes. While the classes *Improvement management, Commitment*, and *Cultural issues* deal with the starting and overall planning of an SPI initiative, the classes *Plan, Do, Check* and *Act* focus on factors that are important when improvement activities are executed. The development process of the CSF criteria is illustrated in *Figure 16*.

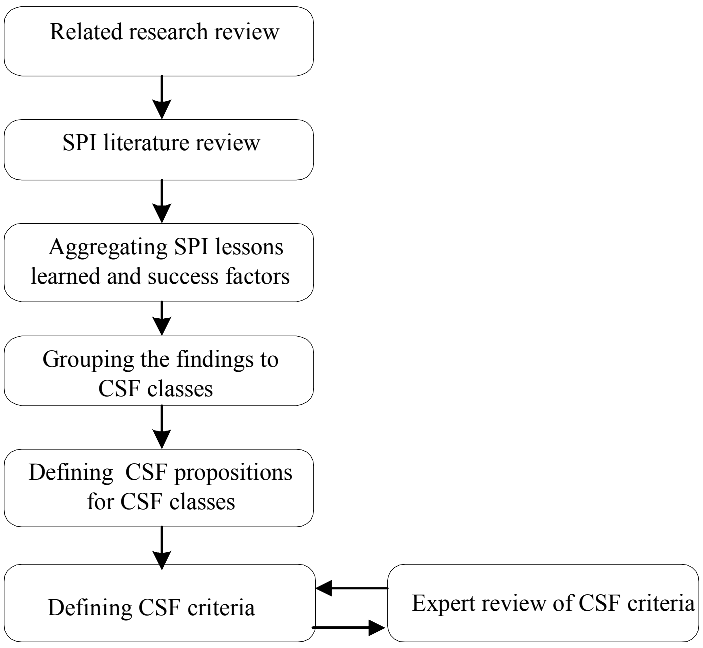
Related research review
SPI literature review
Aggregating SPI lessons learned and success factors
Grouping the findings to CSF classes
Defining CSF propositions for CSF classes
Defining CSF criteria
Expert review of CSF criteria

Figure 16. CSF development process.

3.2 Factors Facilitating SPI

In the following three sections, the results of the literature review are presented. First, 11 SPI case study reports including a total of 34 SPI industrial SPI cases are discussed. These case publications comprise both single case studies (Jakobsen 1998, Delmiglio et al. 1999, Bazzana & Fagnoni 1999, Debou et al. 1999, Simon 1999, Rodenbach et al. 2000) and multiple case studies (Quinn 1996, McGuinness 1999, Lanzerstorfer & Scherzer 1999, Kauppinen & Kujala 2001, Lepasaar et al. 2001). Second, in addition to the case studies, the results of three large surveys targeted to a certain group of organisations are studied: a questionnaire survey conducted by Goldenson & Herbsleb (1995) reporting the results of 56 SW-CMM unique appraisals; a similar survey by El Emam and others (1999) on 14 SPICE improvement cases; and a review by McGuire (1996) of the opinions of 64 software professionals. Last, to augment the review, three publications concerned with a longitudinal involvement in SPI are evaluated (Humphrey 1989, Kunzmann-Combelles 1996, Conradi & Fuggetta 2002).

3.2.1 Industrial Experiences

3.2.1.1 Two Small Companies in the United Kingdom

Quinn (1996) reports the experiences of implementing a quality management system in two small organisations in the United Kingdom. She describes the following success factors for meeting the requirements of ISO 9000 (2000) and ISO9000 based TickIT (TickIT 2001).

- Commitment to improvement. In one of the companies, re-organisations and changes in management set back the improvement program many times;
- Establishment of clear responsibilities and a mechanism for implementing changes in practice in the form of a project;
- Using audits to start the improvement actions;
- Understanding the key role of human resources. The tasks should be occupied with the right persons with the right skills. For example, developers gained early success once they took over their own key practices; and
- Understanding the fact that change is difficult to accomplish. Mistrust and protectionism can be overcome by open discussion, training and awareness sessions conducted, for example, by an external consultant.

When setting the basis for improvement, it turned out that both organisations wanted to develop their own specific kind of approach. One goal of the researchers was to transfer the Quality Management System (QMS) from a more advanced organisation A to organisation B, but it soon became obvious that it would not be just a matter of simple transfer operations – the differences in organisation culture and aptitude for QMS turned out to be too big.

3.2.1.2 Danish Delta

Jakobsen (1998) implemented a process improvement program at Danish Delta Software Engineering. Jacobsen's approach to improvement is quite humane and people centric; accordingly, he has stated that SPI can be a two-way street: improvement can be initiated from the bottom as well as from the top. For the former, management approval is, of course, required. Jacobsen recapitulates his experiences in the following:

- Prepare the field for improvement by personally talking to people and fathom out their aims and wishes towards improvement; this will help planning the future and make people more committed to the forthcoming work;
- Start from the problems, not the solutions;
- Be out there. If a quality-concerned group of people operate just on their own, they easily become isolated from projects. These people need to spread out and provide support to project personnel;
- Plan how to keep the project in continuous improvement progress, e.g. by means of small review sessions held several times per week;
- Conduct reviews, which facilitate spreading out knowledge and promote team spirit;
- Invest on teamwork operating on the "your success is my success" principle; and
- Avoid long textual documents, rather trying to visualise as much as possible.

With the principles described above, successful project improvement was gained. The effort was, however, undermined by an obvious lack of management commitment and resources. Jakobsen refers to this as follows: "Unfortunately, as so often happens in real life, the ideal of theory fell short when applied to the reality of practice. We simply didn't have the time or resources to give these people the proper education, and one team suffered severely from this because they couldn't learn and implement the entire process" (Jakobsen 1998 p. 66).

3.2.1.3 Italtel

Delmiglio et al. (1999) encompass the top twelve lessons learnt from an assessment based SPI programme executed in the GSM application domain at Italtel as follows:

- SPI should utilise different improvement approaches along the different SPI phases;
- The SPI program is best started with an assessment performed by an external actor;
- The SPI program has to be stuffed by SPI experts, middle managers and affected staff members;

- The commitment of senior management has to be 100%, and the support has to be visible;
- The results collected using metrics have to be reported regularly, e.g. monthly;
- Case studies have to be performed before launching the big process and/or technological changes;
- A detailed plan for deployment has to be prepared;
- New technology and tools are often a necessity for a successful SPI effort;
- A training program has to be defined and regularly executed;
- Special attention should be paid when introducing improvements to technical staff, which is proud of its work and not willing to change work procedures for fun;
- SPI should be run faster than the development project, so as not to allow the complexity and product size to override improvement efforts made at project level; and
- The audits performed by a customer boost the improvement actions and help to sustain management commitment.

3.2.1.4 Onion

Bazzana & Fagnoni (1999) report their experiences with SPI as applied at ONION, the Italian Internet Service provider. The authors used the Plan/Do/Check/Act improvement paradigm (Deming 1986) and Bootstrap assessment (Kuvaja et al. 1994) in the planning phase and again in the checking phase, which took place three years after the first assessment. The authors attributed their success to the following three main aspects:

- Involvement of people from different departments;
- Deployment in two pilot projects; and
- Combination of technical and methodological aspects.

For avoiding SPI pitfalls, the authors recommend deploying new guidelines using the bottom-up approach and doing that only under the authorization of the project leader, him or her having discussed and accepted the new guidelines with involved engineers. Here again, the independent and self-respecting role of project managers and engineers is emphasised. In addition, if the pilot project does not show any results, no deployment should take place. Quantitative results play a major role in deciding upon further actions. Bazzana & Fagnoni also stress the significance of internal training, information dissemination, and detailed improvement planning.

3.2.1.5 Alcatel

Debou et al. (1999) report their lessons learnt after several years of SPI experience at Alcatel Italy, another company operating in telecommunication business. Alcatel had started SPI several years earlier and had been using ISO 9001 (ISO 9001 2000), European Quality Award (EFQM 2003) and SW-CMM (Paulk et al. 1994, Paulk et al. 1993) as improvement methods. Based on their long experience with SPI, Alcatel put forward the following issues:

- SPI methods have to be tailorable. According to Alcatel "It is a dream to think that the same improvement approach can be applied everywhere" (p. 283);
- Before launching any major assessment effort, it should be ensured that the organisation is ready for improvement. Debou et al. argue that the assessment process has been greatly overemphasised and that it should be seen only as a starting point, or as "just a tiny part of the iceberg" as they put it (p. 282);
- It is critical to define the strategy for proceeding from assessment results to implementation actions. The experience has shown that if too much time elapses from the assessment to the first impact on projects, the motivation is likely to decrease at all levels;
- The managers' attention can be improved and the improvement speed accelerated by using external consultants in the improvement start-up phase;
- The link to business goals and an active role of business managers are essential for success. A continuous follow-up with metrics is emphasised as well;
- From the methodological point of view, further success factors are to be found in a wise interpretation of ISO 9001 and SW-CMM, and in the willingness for change; and
- The cultural specialities at national and organisation levels need to be understood to be able to speak the same language.

3.2.1.6 CISI Software House

Simon (1999) made his analysis on the basis of his involvement in an improvement program at the CISI software house. The departments of the organisation had been aligned with new business strategies, which changed the structure and composition of software projects. ISO 15504 and ISO 12207 had been selected as input models to support this transformation, which was executed according to the Plan/Do/Check/Act cycle as suggested. Simon concludes his experiences as follows:

- A detailed improvement plan including fixed tasks, schedules and resources needs to be established. This project plan type of plan serves the quality manager as a road map;
- A quality plan or quality requirements need to be drawn out and monitored during the improvement project;
- Operational guidelines for software development projects are very useful and provide a common framework for projects to proceed in practice; and
- Standards provide concepts and good practices, which can be adapted and integrated into organisation-specific process definitions.

3.2.1.7 Five Irish Case Studies

McGuinness (1996, 1999) reports the experiences and lessons learnt of five Irish case studies. The cases cover a large variety of industry sectors: manufacturing (Motorola Manufacturing), financial (Quay Financial Software), insurance (Voluntary Health Insurance) and telecommunication (Tellabs and Telecom Eireann). In their improvement programs, all the organisations except Voluntary Health Insurance applied SW-CMM and all of them some modified form of the Plan/Do/Check/Act cycle. McGuinness recapitulates the feedback from the organisations in the following:

- The role of a sponsor is critical for success. Two strong responsible sponsors were changed during the improvement projects and, despite anticipatory actions, both organisations started to suffer from a decline in improvement. Unless SPI is given strong, visible and active senior management sponsorship with an accompanying vision, SPI is likely to fade and give just moderate results, or even lead to a total failure;

- All the variations of CMM as used by the assessment organisations showed equally good results in initiating the SPI program. The type of assessment may vary, while a clear picture of the starting point is a necessity for SPI;
- The improvement program needs to be guided by an improvement approach or a life cycle. There is no point to proceed without it;
- Improvement does not happen by itself. It has to be planned and tracked as a project with a stretch goal;
- Organisations underline the role of the training, which needs to planned carefully and have clear objectives. Goal-oriented workshop, coaching sessions, short guides, and on-the-job training were found as good training methods;
- Four organisations were deploying the Groupware tools to support process implementation. These organisations pointed out that even though the investment in setting up such an environment requires more resources than estimated and provides less profit than expected, it does have great potential if introduced correctly. Automated support should be provided whenever possible;
- When studying the improvement results, it was found out that large organisations would have to survey all the methods and approaches available on the market before piloting or implementing solutions across the company. Metrics provide help in verifying the improvements and in fathoming out further improvement potentials; and
- Process improvement is not an easy task at all and lots of continuous investment in improvement is needed to keep the progression alive. An overall SPI strategy or an approach is essential for sustaining the obtained level or improving it. It is advisable to have an advance vision of where to go after the initial improvements have been made.

Regarding the question list for SPI success, McGuinness also points out the following factors:

- Start small, step by step, “take a one bite of the elephant at a time” and wait for benefits before extending the improvement areas;
- Ensure strong participation of as many of the people involved as possible. Improvement should be done by people to people. Although the imposed solutions are often strongly opposed, serious actions are appreciated by engineers.

3.2.1.8 Frequents Nachrichtentechnik Gesellschaft M.B.H.

Lanzerstorfer and Scherzer (1999) have combined the ISO 9001 and SW-CMM approaches to a model called BICO (Benchmarking and ISO Combined). The model has been designed for small and middle size organisations that cannot make the investments needed for official audits or assessments. The authors applied this model to nine organisations, and based on their experiences they concluded:

– "Common sense is the most valuable tool for a process improvement expert. If you are an expert, you don't need a quantitative assessment to identify major improvement potentials" (Lanzerstorfer & Scherzer 1999, p. 378).

The case study of Frequents Nachrichtentechnik Gesellschaft M.B.H. was described in detail. The company operates in air traffic control business and shows a very strong management commitment. At the time of the study, the company had held the ISO 9001 quality certification for five years. Based on this study they conclude that:

– Management commitment and strong project management are key success factors; and
– Distribution of the information of what is going on facilitates buy-in.

3.2.1.9 Tokheim

Rodenbach et al. (2000) describe their ten years of experience with SPI at Schlumberger RPS, which later became part of Tokheim. In this project, the authors used various improvement approaches: ISO 9001/TickIT, SW-CMM, Bootstrap, Goal/Question/Metric, Spirits, and later also the Profes methodology. The essential factors are described as follows:

– Commitment is crucial and should be present at all levels, including high-level management, project management and software engineers;
– Management commitment is often sought at the beginning of the improvement actions, but it should be addressed continually. They learnt real risk for improvement programs is to be found in the changes in management; after a change it takes a while before the commitment is re-established. The even experienced a fallback in process maturity when they did not take these changes seriously enough;

- Commitment of software engineers is important as well, and often underestimated too. There is a big risk involved when fully-fledged solutions from standards and best practices are imposed on engineers. Based on their experiences the authors claim that the acceptance and commitment of engineers is achieved by introducing improvements that are based on their own ideas. “Treat software engineers as what they are: intelligent creative professionals. Let them define their own goals for improvement” (Rodenbach et al. 2000, p. 227);
- Practical support offered by an internal support group is essential. Support should be provided in a form of implementation assistance, coaching all the way through the project, in gathering measurement data and analysing it and in assuring management commitment; and
- Successful SPI programs cannot be transferred as such from another culture. Different countries, companies, types of site, all form different cultural entities.

3.2.1.10 Two Finnish Organisations

Kauppinen & Kujala (2001) reported their experiences of requirements engineering process improvement in two Finnish organisations. As an improvement approach they used selected parts of the IDEAL model and the ISO 15504 standard. The researches have noted that often organisations do not have enough resources or expertise to use such sophisticated approaches.

The case companies operate in the area of real-time embedded software and interactive software development. The following SPI success factors were stated:

- Setting of measurable goals;
- Ensuring management support;
- Use of improvement teams formed of experienced practitioners; and
- Capability to select realistic improvement actions.

According to the researchers, change may require a culture change in addition to changes in process and technology. The change in culture requires that the personnel understand the reasons for the change.

3.2.1.11 Ten Small Finnish Organisations

Lepasaar et al. (2001) have been involved in a regional SPI network initiative called SataSPIN consisting of over 10 small Finnish software organisations. These organisations provided a prioritised list of the most important SPI factors:

- SPI related training;
- External guidance of the SPI work;
- Company's commitment to SPI activities;
- External support for SPI;
- SPI environment support for a sufficiently long period of time (external mentoring);
- Availability of information about SPI;
- External financial support;
- Availability of company's own resources; and
- Measurable targets set for SPI work.

Based on their analysis and comparison of SataSPIN organisations the researchers named four main factors affecting SPI in small organisations:

- Public funding to start the initiative (particularly in case of small organisations);
- Learning environment including the means of knowledge acquisition and training and a mentor that could be an external consultant to provide help in assessments and improvement activities;
- Readiness to invest resources in SPI; and
- Establishment of continuity in the SPI work.

3.2.1.12 Summary of the Cases

The experiences of 34 industrial SPI cases are summarised above. For each case, a body of success factors and lessons learnt were pointed out. It can be concluded that almost all the cases used more than one SPI method, most often a combination of two methods, one of these methods often being an assessment method (SW-CMM, ISO 15504 or Bootstrap). ISO 9001 is also used regularly as a guideline for improvement. The raised success-related issues soon started to be refined into statements. When no more new factors seemed to be found, the literature review of SPI cases was finished. The success factor cited most often was the need for commitment at various organisational levels. The found success factors groups and the number of related statements are presented in the end of this Chapter.

3.2.2 Surveys to Detect SPI Success Factors

3.2.2.1 Extensive SEI Survey

A comprehensive study conducted by SEI (Goldenson & Herbsleb 1995) surveyed possible SPI barriers and success factors. The survey took place in 1994 and it was targeted at organisations in the United States and Canada, where SW-CMM assessment had been carried out during the years 1992 and 1993. The sample consisted of a total of 138 filled questionnaires representing 56 unique appraisals. The largest number of answers, 37%, came from US government contractors, 22% from the federal government and military services. Organisations with their own product development sections were represented by 36% of the replies. Based on the survey SEI summarised the success factors as follows:

- SPI monitoring by senior management;
- General awareness of SPI goals;
- Senior management understanding of technical issues;
- Respect of SPI personnel;
- Technical staff involvement in SPI;
- Willingness of management to take risks; and
- Clear, compensated SPI assignments.

In addition to success factors, barriers were also studied by SEI, and as a result the following factors were found to be of substantial or moderate importance:

- SPI gets in the way of “real” work;
- Paperwork required;
- Organisational politics;
- Reorganisation/staff downsizing;
- Discouragement about SPI prospects;
- Senior management turnover; and
- “Turf guarding” inhibiting SPI.

3.2.2.2 Large Survey within an American Organisation

McGuire (1996) conducted a three-phase survey in a large American software development project. The survey comprised 64 software professionals representing various positions in the organisation. The research was concerned with, among other things, the success factors for SW-CMM based SPI. From the survey results the following key issues were obtained:

– Organisational culture and related change management strategy coupled with appropriate training and information sessions can have a decisive effect on the rate of improvement progress;
– Specific, just-in-time training and information enhance team performance;
– Increased emphasis on team-based, quality-focused, process-dependent organisational culture and structure is likely to pay off; and
– Feedback loop and facilitation are factors that enable successful process improvement, also sustaining it.

McGuire also makes the remark that, in the context of SW-CMM-based SPI, it is unlikely that an organisation will understand all the aspects and needs of a large and long-term improvement project.

3.2.2.3 Results of SPICE trials

El Emam et al. (1999) made a questionnaire survey of 14 SPI cases to find out what the success factors affecting assessment based improvement when ISO 15504 was used as an improvement method. The authors drew upon an earlier survey on SW-CMM appraisals (Goldenson & Herbsleb 1995), which had been executed by the Software Engineering Institute, and adapted the questions to their own project. It was found out that the awareness and understanding of SPI goals and technical staff involvement in SPI are among the most critical success factors for assessment based SPI efforts. The essential SPI factors are:

– SPI monitoring by senior management;
– Compensated SPI responsibilities;
– Staff and time resources made available for SPI; and
– High respect of SPI people.

All the factors mentioned above should increase the likelihood that the SPI effort is correctly determined by the initial assessment findings, which was interpreted to be one of the cornerstones of SPI success. The SEI survey had came to the same conclusion four years earlier.

3.2.3 Arguments for Successful SPI

SW-CMM (Paulk et al. 1993, Paulk et al. 1994) has become the most widely used method in SPI; its ideas were initially introduced by Humphrey (1989). The author of the SW-CMM model came up with a list of six basic principles for software process change (Humphrey 1989, p. 19):

- Major changes to the software process must start at the top. Senior management leadership is required to launch the change effort and to provide continuing resources and priority;
- Ultimately, everyone must be involved. Software Engineering is a team effort, and anyone who does not participate in improvement will miss the benefits and may even inhibit progress;
- Effective change requires a goal and knowledge of the current process. To use a map, you must know where you are;
- Change is continuous. SPI is not a one-shot effort; it involves continual learning and growth;
- Software process changes will not be retained without conscious effort and periodic reinforcement; and
- SPI requires investment. It takes planning, dedicated people, management time, and capital investment.

As a supplement, Kunzmann-Combelles (1996), based on her involvement in various SPI cases, complements Humphrey’s list by stating that the appraisal model is not as important a factor as business goals, which are considered the key driver of the improvement program. She expands the leadership role from senior management to middle management and also stresses the need of effectiveness measurement. She claims that if the above-mentioned criteria are not satisfied, there will be no successful SPI.

In their recent publication, Conradi & Fuggetta (2002) put forward six SPI requirements based on their involvement on SPI:

- SPI framework should support improvement strategies focusing on goal-orientation and product innovation;
- Developers are motivated for change; if possible, start bottom-up with concrete initiatives;
- Automated software process support has been overemphasised;
- Cost-benefit analysis requires novel amortisation models;
- SPI assumes cultural changes, so we need expertise from the social sciences; and
- SPI is about learning – not control, as in QA.

While SPI initiatives need to be directed towards business goals, they should also start from the most pressing needs of organisation or projects. At the same time as improvement goals have to be realistic and visible, the improvement has to start with small steps, one well-defined SPI effort at a time. An SPI initiative should not be started with a large assessment, but rather by using a simple and focused scorecard. The role of mid-managers in SPI is essential, too. Too high expectations of the engagement party should be dampened because of the fact that achieving convincing and repeatable results may take some time. Conradi & Fuggetta emphasise the role of the cultural, learning, and long-term dimensions of SPI work and promote establishing participative engagement with all process changes. The authors even propose using multidisciplinary improvement teams for performing empirical studies on how people actually work. To motivate people for continuous process improvement, a rewarding system for reported problems and suggested improvements is considered important. As it is generally difficult to demonstrate the return of investment in SPI, the need for developing organisation-specific cost-benefit models is raised. (Conradi & Fuggetta 2002).

3.2.4 Summary of SPI Success factors

The SPI success factor statements as obtained from the SPI literature review are presented above. These statements are organised into seven statement classes. The classes improvement management, commitment, and cultural issues are developed based on the author's statement evaluation and synthesis. The PDCA cycle is used for organising the success factors concerning SPI engineering itself.

The results show that the overall quality of SPI management is the most decisive factor for success (36% of all statements). *Figure 17* illustrates how the statements are distributed over the success factor classes. It can be concluded that the success factors related to improvement management and planning are greatly emphasised in the literature. It is also interesting how the improvement cycle itself (Plan-Do-Check-Act) is stressed: 59% of all remarks concern the starting phase of the initiative. Least attention is paid to the period when the improvement project should shift from planning phase to piloting or taking the results in use in large scale.

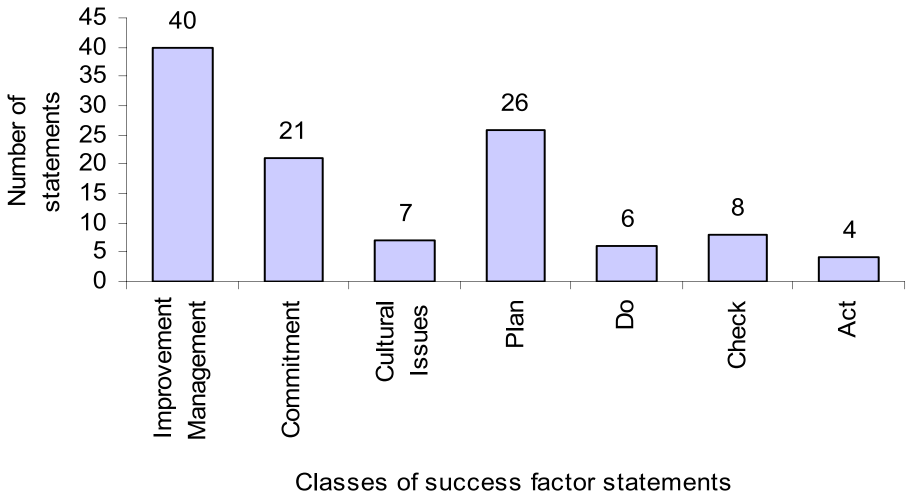

Figure 17. Success factor statements by classes.

Judging by the number of statements, three areas rise above the others: improvement management, commitment, and the plan phase. The improvement management success factor area consists of general guidance, training, and staffing issues. Commitment as a success factor area appears highly solid and unequivocal thus highlighting its meaning. The thirdly critical activities for improvement success are to be found in careful and intelligent planning phase execution, which is when current state is analysed, improvement goals are set, and a concrete improvement plan is developed.

In *Table 7*, the success factor statements are divided into seven main classes and eight sub-classes. Appendix B presents all statement items grouped to seven main classes.

Table 7. SPI success factors as observed in literature.

Main Classes	*Sub-classes*		Number of statements
1. Improvement Management			40
	General guidance	*19*	
	Staffing the SPI initiative	*13*	
	Training	*8*	
2. Commitment			21
	Manager commitment	*17*	
	Engineer commitment	*4*	
3. Culture			7
4. Plan			26
	Current state analysis	*8*	
	Goal definition	*11*	
	Improvement planning	*7*	
5. Do			6
6. Check			8
7. Act			4
Total			112

In the following sections, the success factor classes and statements are studied by the developed CSF groups. Based on the evaluation of the original success factor statements, the requirements for SPI methods are formulated into propositions, and the result is called the CSF criteria.

3.3 Evaluation of CSF Classes

3.3.1 Improvement Management

According to a large body of researchers, competent SPI management is essential for the success of SPI activities (Humphrey 1989, Goldenson &

Herbsleb 1995, McGuinness 1996, McGuinness 1999, McGuire 1996, Kunzmann-Combelles 1996, Quinn 1996, Jakobsen 1998, Bazzana & Fagnoni 1999, Debou et al. 1999, Delmiglio et al. 1999, El Emam et al. 1999, Lanzerstorfer & Scherzer 1999, Kauppinen & Kujala 2001, Lepasaar et al. 2001, Conradi & Fuggetta 2002). Even though this success factor may appear self-evident, it is not obvious, in the context of SPI, what good management means in practice. Based on the literature evaluation, three associated success factor sub-classes for improvement management were distinguished: general SPI guidance, staffing of the improvement initiative, and training.

3.3.1.1 General SPI Guidance

This class contains a number of various success factors such as "Management should be willing to take the risk that is always related to change" (Goldenson & Herbsleb 1995), "Senior management should monitor SPI" (Goldenson & Herbsleb 1995, El Emam et al. 1999), and "Understand technical issues" (Goldenson & Herbsleb 1995). According to Jakobsen (1998), the software quality group should not isolate itself from projects, but spread out and provide hands on support. Debou et al. (1999) maintain that the readiness of an organisation for improvement should be ensured before SPI actions could take place. Debou et al. also argue that the management should understand the fact that software process assessment is only the starting point for SPI. McGuinness (1996, 1999) recommends for large organisations to survey the methods and approaches available on the market before piloting or implementing solutions, while other researchers counsel to dampen any overly positive expectations of all parties involved, as attaining convincing results may take some time (Conradi & Fuggetta 2002).

Those who have been using various improvement models or approaches in SPI recommend this as a general success factor. Delmiglio et al. (1999) contend that an SPI effort should utilise different improvement approaches along the initiative phases of SPI. Debou et al. (1999) suggest that SPI methods should be tailorable, and they further argue that a single improvement approach cannot be applied everywhere and that a sound strategy is required for proceeding from the assessment results. McGuinness (1996, 1999), for his part, makes a general remark concerning the importance of the improvement approach or the life cycle as a backbone of the improvement program. The author also argues that any variation of the SW-CMM assessment showed equally good results. The latter observation is stated also by Kunzmann-Combelles (1996), who claims that the model used for appraisal is not as important as the role of business goals. These statements advocate a broadminded approach and wise interpretation of models within SPI.

In the first paragraph of the general guidance discussion section, it was not possible to define any general success factor requirements due to the lack of uniformity among the statements. However, from the success factor discussion regarding improvement models presented above, the following general SPI proposition was captured:

CSF-1: Capability to include or support different SPI approaches.

CSF-1 description:

CSF-1 means that the SPI method used for managing the SPI initiative should not only determine the main phases of the improvement initiative but also to recognise and point out opportunities for using various approaches, methods, or techniques along the improvement process.

Characteristics of an SPI method supporting CSF-1:

- Presents an overall improvement path and philosophy to follow,
- Refers to improvement approaches, methods or techniques that may be utilised during the course of the improvement work,
- No predetermined set of improvement techniques are set,
- No predetermined set of improvement targets are set.

3.3.1.2 Staffing the SPI Initiative

It should have a great importance how an SPI initiative is manned. A large SW-CMM study by Goldenson & Herbsleb (1995), which was later repeated for SPICE trials (El Emam et al. 1999) asserts the following staffing factors to be cornerstones for success: SPI people should be well respected; and SPI assignments should be compensated and technical staff should be involved in SPI. The importance of technical personnel participation, in particular, which was expressed in several forms such as involvement of "experienced practitioners", "people from different departments" or "affected staff members", was strongly promoted by other researchers as well (Humphrey 1989, Delmiglio et al. 1999, Bazzana & Fagnoni 1999, Kauppinen & Kujala 2001). Quinn (1996) argues on behalf of active participation of staff by stating that once developers took over their own processes early success was gained. The Irish SPI study including five organisations reports similar experiences; imposed solutions are strongly opposed by software engineers (McGuinness 1999).

From the above statements, two propositions for successful SPI may be put forward:

CSF-2: Active participation of all affected parties.

CSF-2 description:

This means that active participation denotes taking part in developing improved practices and, on the other hand, reviewing and commenting proposed new practices. Affected party refers to persons or organisational units whose practices will be influenced by improvement changes. Affected parties may thus include line managers, product managers, quality managers, and representatives of pilot projects or project groups. The group of affected parties should also vary during the lifecycle of an improvement program (a smaller pilot project versus institutionalising new practices).

Characteristics of an SPI method supporting CSF-2:
- Affected parties are consulted when selecting improvement goals,
- Affected parties are consulted when developing improved practices to achieve the goals.

CSF-3: Co-operation with software engineers.

CSF-3 description:

This means that software engineers are concretely involved in actual improvement work stating their needs and ideas for improvement, actively taking part in change planning, and providing feedback.

Characteristics of an SPI method supporting CSF-3:
- Software engineers are encouraged to take part in improvement planning,

- Software engineers' feedback to plans is sought and regarded as important and valuable.

3.3.1.3 Training

Training is an important management asset when heading for successful SPI (McGuire 1996, Quinn 1996, McGuinness 1999, Delmiglio et al. 1999, Bazzana & Fagnoni 1999, Lepasaar et al. 2001). While some researchers advocate specific, just-in-time training (McGuire et al. 1996), others underline the meaning of careful planning of training with clear goals in mind and regularity of training sessions (McGuinness 1999, Delmiglio et al. 1999). Quinn (1996) points out that training is a mean to overcome mistrust and protectionism. Those researchers who regard training as a success factor argue strongly for it. In view of the fact that mentoring or practical support is essential in a form or another, Rodenbach et al. (2002) argue for setting up an internal support group, while Jakobsen (1998) reports weekly sessions as a mean to sustain the improvement rhythm.

The following training related SPI proposition can be generalised from the above statements:

CSF-4: Training is planned and made part of the initiative.

CSF-4 description:

This means that training needs are clarified, the content, schedule and participants are planned, and training is provided for any new practices. Rather than providing general level SPI motivation presentations, training should be case specific and repeatable, in view of the fact that the use of new practices is likely to expand in an organisation.

Characteristics of an SPI method supporting CSF-4:
- Training needs are clarified and training plan is developed,
- SPI facilitator or support person is assigned to assist software engineering projects in practice.

3.3.2 Commitment

Commitment is strongly emphasised by many researchers, there is no argument against its great significance to the outcome of the SPI effort (Humphrey 1989, Quinn 1996, Kunzmann-Combelles 1996, McGuinness 1996, McGuinness 1999, Jakobsen 1998, Delmiglio et al. 1999, Lanzerstorfer & Scherzer 1999, El Emam et al. 1999, Rodenbach et al. 2000, Kauppinen & Kujala 2001, Lepasaar et al. 2001, Conradi & Fuggetta 2002, Abrahamsson 2002). Opinions diverge, however, when it comes to the focal point of commitment. Humphrey (1989), representing the SW-CMM assessment based top-down SPI approach, consequently stresses the commitment of senior leadership and top managers. Kunzmann-Combelles (1996), again, argues for expanding this perspective to include middle management as well. In the late 1990's, the first strong statements were presented advocating strong engineer commitment (Jakobsen 1998, McGuinness 1999, Rodenbach et al. 2000). Many researchers see that SPI benefits are best acquired through ensuring a strong team or project level commitment. Rodenbach and others (2000, p. 227) claim that "Commitment of software engineers is important as well, and often underestimated too", and continue "We experienced that the best way to get acceptance and commitment of engineers is to introduce only improvements that are based on their own ideas". Top management commitment seems to be crucial in launching and sustaining an SPI initiative, allocating resources and providing support. In addition to this, the desired outcome is better achieved if software engineers are committed to the change. Despite increased engineer centrism, commitment still cannot be seen to be formed solely with a bottom-up approach; it is more likely to come about in a well-balanced top-down and bottom-up game plan.

On the basis of above discussion, the following three propositions may be extracted:

CSF-5: Commitment of top managers.

CSF-5 description:

This means that top managers have to be motivated and actively made aware of the importance of the improvement initiative and the gained results. Commitment has to be re-ensured intermittently to ensure adequate resources for SPI in the future.

Characteristics of an SPI method supporting CSF-5:

- Top managers are consulted and made aware of justified improvement actions,
- Top managers are informed of the status and results of the work.

CSF-6: Commitment of middle managers.

CSF-6 description:

This means that when it comes to resources, middle managers often play a similar role as top managers, the impact of middle managers often being the more important the bigger the organisation.

Characteristics of an SPI method supporting CSF-6:

- Middle managers are consulted and made aware of justified improvement actions,
- Middle managers are informed of the status and results of the work.

CSF-7: Commitment of software engineers.

CSF-7 description:

This means that special attention has to be paid to making software engineers committed to SPI. Researchers and practitioners argue this is best ensured by actively involving engineers in determining the required new practices.

Characteristics of an SPI method supporting CSF-7:

- Software engineers are encouraged to take part in improvement planning,
- Software engineers are consulted and made aware of justified improvement actions,
- Software engineers' feedback to plans is sought and regarded as important and valuable,
- Software engineers are consulted for the status and results of the improvement actions.

3.3.3 Cultural Issues

Several studies underlined the importance of being aware of the differences between the various national, organisational and site level cultures affected by

SPI (McGuire 1996, Debou et al. 1999, Rodenbach et al. 2000, Kauppinen & Kujala 2001, Conradi & Fuggetta 2002). This means that neither SPI solutions nor programs can be transferred successfully as such. The specific cultural features need to be understood to be able to speak even the same language (Debou et al. 1999). Based on a survey on 64 software professionals McGuire (1996) binds cultural aspects with change management strategies and training, and reasons that if put together these may have a substantial effect on the rate of the improvement progress. Kauppinen & Kujala (2001) propose that SPI calls for a cultural change, and they go on to argue that, basically, cultural change requires that the personnel understand the reason for the change. To alleviate the difficulty of cultural transformation, Conradi & Fuggetta (2002) propose that SPI should even utilise expertise from the social studies. As it is ineluctable that culture differs from organisation to organisation, and from country to another, it can be understood that ready wrapped solutions are bound to be insufficient and thus also likely to cause opposition.

The culture-related observations and statements led to the following SPI proposition:

CSF-8: Improved solutions are developed on a case-by-case basis.

CSF-8 description:

This means that even if some well-known best practices are used as a basis to improve processes they always have to be adapted to the specific use. Consequently, the needs, characteristics and culture of the organisation and projects in focus have to be detected and understood.

Characteristics of an SPI method supporting CSF-8:
- Organisational and project characteristics and culture form the basis for developing improved solutions,
- No ready made resolutions are introduced or transferred as solutions.

3.3.4 Plan

Planning-related success factors are divided into three categories according to their content: current state analysis, goal definition, and improvement planning.

3.3.4.1 Current State Analysis

Humphrey established the basic success factor for any improvement initiative in the following words, which have been eagerly cited by others ever since: “To use a map, you must know where you are” (Humphrey 1989, p. 19). Even though there is unanimity on this factor, i.e., that the current status of processes must be fathomed out in order to be able to set any meaningful improvement goals, opinions vary on how to determine the position on the map. Although assessment-based approaches such as SW-CMM or ISO 15504 rely solely on assessment results, they accept various approaches to conducting the assessment (McGuinness 1999, Delmiglio et al. 1999). Still, assessment is not considered as the only mean to start the improvement; audit, for example, has been proposed as an effective approach, too (Quinn 1996). Furthermore, even the simple and focused scorecard method has been recommended instead of extensive assessments (Conradi & Fuggetta 2002). Lanzerstorfer & Scherzer (1999, p. 378), for their part, rely on expertise, arguing that “Common sense is the most valuable tool of a process improvement expert. If you are an expert, you don’t need a quantitative assessment to identify major improvement potentials”. Conradi & Fuggetta (2002) have moved down a long way from the top-down SPI approach to grass-roots level: They put out the flag for using multidisciplinary improvement teams to perform empirical studies on how people actually work and also advocate starting improvement bottom-up with concrete initiatives.

Based on the success factor discussion above, the following SPI proposition can be presented:

CSF-9: Current status of processes is clarified.

CSF-9 description:

This means that an understanding of current software development process and its weaknesses and strengths is established.

Characteristics of an SPI method supporting CSF-9:
- Current software development process is evaluated,
- The weaknesses and strengths of current practices are known.

3.3.4.2 Goal Definition

Goal definition presents a critical improvement phase since prudently selected improvement goals build the backbone for all further actions within the SPI initiative. Improvement goals have to be realistic and visible (Conradi & Fuggetta 2002), measurable (Kauppinen & Kujala 2001, Lepasaar et al. 2001) and well understood (Goldenson & Herbsleb 1995, El Emam et al. 1999). Kauppinen and Kujala further claim that the success of an SPI program relies on the capability to select realistic improvement actions. The importance of a link from business goals to improvement goals is stressed as well (Kunzmann-Combelles 1996, Debou et al. 1999, Conradi & Fuggetta 2002). Successful improvement initiative is not an isolated action but linked to the overall goals of the organisation. Jakobsen (1998) proposes starting from the problems, not from the solutions. This is recommended by Conradi & Fuggetta (2002) as well, who suggest starting from the most pressing needs of the subject company or project. The authors also advocate for starting small, one SPI effort at a time, as is also suggested by McGuinness (1996).

From this discussion it may be concluded that an SPI method should fulfil the following two propositions:

CSF-10: Link between business goals and improvement goals is established.

CSF-10 description:

This means that the selected improvement goals should support the achievement of business goals and the link between the goals is defined and visible.

Characteristics of an SPI method supporting CSF-10:
- Business goals are clarified and taken into consideration when improvement goals are set,
- How the achievement of an improvement goal supports the satisfaction of a business goal is addressed.

CSF-11: Improvement goals are measurable.

CSF-11 description:

This proposition refers to the issue that without setting measurable improvement goals it is difficult to justify whether SPI has been successful or not. Measurable goals set a basis for selecting metrics for evaluating the effects of actions. Initially, there was a competing proposition for measurable improvement goal setting: “the improvement goal needs to be realistic and well understood”. Even though this statement is valid and important from a SPI success point of view, it was not selected due to the difficulties concerning the capability of SPI methods to support this proposition, and also due to its problematic evaluation. The prowess of setting realistic goals refers more to the capability of SPI personnel. The statement of improvement goals being well understood is rather a vague one, though this type of knowledge creation is an inbuilt issue in many CSF propositions, such as CSF2 “Active participation of all affected parties” or CSF4 “Training is planned and made part of the initiative”.

Characteristics of an SPI method supporting CSF-11:

- Improvement goals are quantitative in nature, and indicative of wanted results.

3.3.4.3 Improvement Planning

Even though the features of improvement planning as a success factor fail to attract any major attention, the important role of detailed improvement planning including task, schedule and resource planning is addressed by a number of researchers (Delmiglio et al. 1999, Bazzana & Fagnoni 1999, Simon 1999, McGuinness 1996). This plan is called an improvement plan or a quality plan. Based on five case studies, McGuinness (1999, p. 335) puts the lessons learnt briefly in the following: “Improvement doesn’t happen by accident. It has to be planned.” He further points out that improvement has to be planned and tracked as a project and the results should be at hand before extending improvement areas.

To form the basis for a successful execution of an improvement initiative,

CSF-12: An improvement plan is generated.

CSF-12 description:

This proposition means that the tasks, schedule, resources, reporting, and follow-up have to be planned and documented.

Characteristic of an SPI method supporting CSF-12:
- A practical improvement implementation plan is generated.

3.3.5 Do

The previously developed plans are implemented in this phase to achieve the desired improvement goals. It is more certain to execute the take up phase successfully if it is started with case studies or pilot projects before launching major changes (Delmiglio et al. 1999; Bazzana & Fagnoni 1999). Furthermore, the deployment is recommended only if the project manager is in charge of the changes and new guidelines are mutually agreed upon by project managers and software engineers. Besides, in a situation where the project manager has been authorised to execute the software project as she or he sees most reasonable this is the only way to succeed. Operational guidelines are named as one practical success factor by Simon (1999), referring to detailed process instructions as to what to do in the different phases of a software development project. Debou and others (1999) point out that external consultants may accelerate the improvement speed.

From this slightly discursive discussion the following proposition may be extracted:

CSF-13: Developed solutions are tested before large-scale use.

CSF-13 description:

This means that improved solutions are tested and validated in a pilot project or projects before proposing them for a large-scale use in an organisation.

Characteristics of an SPI method supporting CSF-13:
- Pilot project is selected, or other case studies are planned,
- Improved practices are developed taking into account pilot project needs.

3.3.6 Check

Monitoring and following up of improvement actions using metrics is advocated by several authors as critical actions in the process improvement endeavour (Kunzmann-Combelles 1996, Debou et al. 1999, McGuinness 1999, Conradi & Fuggetta 2002, Rodenbach et al. 2000). McGuinness (1999) states explicitly that metrics provide help in the verification of improvements and in addressing possible improvement areas. Similarly, Debou et al. (1999) argue strongly for quantitative results; if, for example, a pilot project shows no improvement, no deployment should take place. Rodenbach and others (2000) consider measurement vitally important to SPI and recommend easing the overhead of a project by assigning the measurement gathering and analysis tasks to a support group. Conradi & Fuggetta (2002) approach checking from the "return of invest" point of view and raise the need to establish novel approaches to calculating company-specific cost-benefits. Delmiglio et al. (1999) suggest paying special attention when showing any results to software engineers.

The following SPI proposition can be extracted from the above statements:

CSF-14: Improvement actions are regularly monitored using metrics.

CSF-14 description:

This means that a measurement plan for monitoring improvement actions is developed and used as a tool during the improvement initiative and also afterwards when analysing the results. The essential prerequisite for this is that the improvement goals are measurable.

Characteristics of an SPI method supporting CSF-14:
- A measurement plan is generated,
- Measurement data is analysed during the course of the improvement initiative.

3.3.7 Act

How to successfully keep SPI in continuous action? Event though this difficult question is faced by all SPI initiatives, it was the one receiving the least attention in related literature. Humphrey (1989) is concerned about continuous improvement and points out that changes will not be retained without serious involvement and periodic reinforcement. Lepasaar et al. (2001), again, simply makes an argument for establishing continuity in SPI work. McGuire (1996), for his part, regards the existence of feedback loop and facilitation as further success factors. McGuinness (1999), finally, emphasises the role of metrics in clarifying further improvement potentials, thus considering metrics as a means of sustaining improvement.

Consequently, the following general SPI proposition can be formulated:

CSF-15: Sustainability of an improvement initiative.

CSF-15 description:

This means that a plan to sustain improvement is generated and agreed on. The actions for sustaining the improvement initiative may be planned in connection with improvement planning and willingly linked with the higher level improvement or organisational plans, e.g., with the business plan of the subject organisation.

Characteristics of an SPI method supporting CSF-15:
- A master plan is developed that relates individual improvement initiatives to larger entities.

3.4 The CSF Criteria

The success factor propositions extracted from the literature survey are recapitulated in *Table 8*. The CSF criteria involve 15 propositions, which are grouped to seven classes of similar statement types. Three of the classes are topics that enable or support successful improvement work (improvement management, commitment, and cultural aspect), while four classes are formed according to the generally accepted problem solving process model (plan-do-check-act).

These qualitative CSF criteria are used for evaluating whether an SPI method satisfies these propositions. The assumption is that the more propositions are supported the better the possibilities for success. However, it is also acknowledged that process improvement requires expert knowledge, which can never be substituted by any method. Still, an SPI method can guide improvement actions in a way that helps the project succeed. Appendix A presents the propositions in a form of questions, for which answers are sought by the SPI method evaluation. For the sake of simplicity, the CSF criteria use the “yes - no” measurement scale.

Table 8. The factors for a successful SPI Initiative.

Main Classes – *Sub Classes*	Success factor propositions
1. Improvement Management	
– *General Guidance*	1. Capability to include or support different SPI approaches.
– *Staffing the SPI Initiative*	2. Active participation of all affected parties.
	3. Co-operation with software engineers.
– *Training*	4. Training is planned and made part of the initiative.
2. Commitment	5. Commitment of top managers.
	6. Commitment of middle managers.
	7. Commitment of software engineers.
3. Cultural Issues	8. Improved solutions are developed on a case-by-case basis.
4. Plan	
– *Current State Analysis*	9. Current status of processes is clarified.
– *Goal Definition*	10. Link between business goals and improvement goals is established.
	11. Improvement goals are measurable.
– *Improvement Planning*	12. An improvement plan is generated.
5. Do	13. Developed solutions are tested before large-scale use.
6. Check	14. Improvement actions and results are followed regularly using metrics.
7. Act	15. Sustainability of improvement initiative.

3.5 Validation of CSF Criteria

The evaluation of the validity of the CSF criteria is based on the measures set by the grounded research theory. Glaser (Bryant 2002) names four criteria which should be met by the grounded theory, i.e. by the CSF criteria, by stating: "They should have *fit* and *relevance*, they must *work* and be *modifiable*" (Bryant 2002, p. 256). The criterion *fit* is met thanks to the CSFs being developed based on the success factor data. Expert opinion was used for evaluating the developed CSF criteria; the CSFs were collegially assessed to be a valid embodiment of factors for successful SPI. When a theory *works*, it explains what has happened, interprets what is happening, and predicts what will happen. The CSF criteria condense unique success factor statements to 15 general principles which SPI initiatives have shown in the past when improvement success has been achieved. The CSF criteria are based on the assumption that the better the SPI initiative is able to cover the CSFs the better the possibilities are for improvement success. Therefore, the CSF criteria adequately fulfil the criterion of working theory. A theory is *relevant* if it fits and if it works. Consequently, it can be stated that the CSF criteria are relevant. The requirement of *modifiability* is also met: as new data is collected and analysed, the emerging theory, thus the CSF criteria, can be modified accordingly when needed.

In addition to the evaluation of validity based on the grounded theory, CSF criteria validation is carried out as proposed by (Straub 1989). Measurement instrument validation consists of three components: content validity, construct validity and reliability. *Content validity* means that the instrument contains representative questions of all possible questions that could be stated. The development of CSF criteria followed the general recommendations given by Straub (1989). After an extensive literature review, the 7 CSF classed were defined and the criteria were reviewed and commented by SPI experts. Using this body of SPI knowledge it can be argued that the measurement instrument has content validity. *Construct validity* is an operational concept that clarifies whether the developed instrument measures what it should measure. This is a relevant assumption as the CSF criteria are developed using existing SPI success factor data. *Reliability*, again, refers to the stability of the results retrieved using the instrument. The use of CSF criteria is based on a qualitative analysis of the material available on SPI methods. To enhance the reliability of the CSF criteria, method characteristics supporting each of the CSF criteria are developed.

The findings of other researchers support the developed CSF criteria. Dybå (2000) found 6 main SPI success factors using a literature review and a questionnaire survey. Of these factors, 5 overlap with the defined CSF criteria (business orientation, leadership involvement, employee participation, concern for measurement, and exploitation of existing knowledge) and 1 is partially covered by the CSF criteria (exploration of new knowledge). Another very recent study of critical SPI success factors by Niazi et al. (2004) highlighted issues concerning improvement management and commitment. Based on a literature review and an empirical study, the authors list eight critical success factors most often raised: senior management commitment, staff involvement, staff time and resources, experienced staff, SPI awareness, formal methodology, training and mentoring, creating process action, and reviews. Of this list of factors, all the defined CSF criteria were covered with the exception of review, which failed to be raised by any of the references used here.

3.6 Discussion

In this chapter the CSF criteria are established. The CSF criteria are based on the analysis of the success factors of industrial SPI cases, results of surveys revealing improvement facilitators and expert opinions. Based on the evaluation, seven main success factor classes and 15 related propositions are defined. The resulting CSF criteria are utilised later in this research in assessing how well an SPI method supports the facilitator factors that are found to be important in industry. The possible interdependences of propositions are not studied due to the fact that the available review material did not provide adequate enough information for the purpose.

In the CSF criteria, all the success factors are presented as if they should exclusively relate to a single method, which, however, is not the case. The CSF criteria, essentially, name factors that generally support the achievement of improvement success. CSFs may be fulfilled by several means: by compiling various methods supplementing each other, by addressing organisation level activities (for example CSF 2, 3, 4, 5, 6, and 7), while some may even prove to be irrelevant in certain specific improvement contexts. For example, if the organisation is small, no middle managers may exist, which will make CSF 6 irrelevant in that specific improvement context.

The CSF criteria are used in the following chapters as a means of evaluating related research and also the results of this research.

4. Evaluation of Related Research

In this chapter the developed CSF criteria set is used for evaluating related research. The purpose is to assess how well SPI methods are able to support the critical success factors arisen form the notifications of the industry. As software process and product quality standards are addressed in this evaluation only briefly, they do not fall into the category of SPI methods, but yet are relevant elements of SPI in the context of this thesis. The principles of related research are studied by the author as they are generally presented in the literature, and compared to the CSF propositions. The purpose of this qualitative and subjective study is to gain an understanding of how well these methods support the areas found essential for improvement success.

4.1 Evaluation of Related Research

4.1.1 SPI Management Methods

SPI management methods guide the SPI initiative by pointing out what to do and in which order.

When evaluating the PDCA Control Cycle as presented by Ishikawa (1985), it can be argued that the PDCA cycle is the best choice for addressing critical success factors. It is also the only method in its class to extend commitment verification to the engineer level. However, the co-operation with engineers in solution development is not clearly stressed. The deficiencies of the PDCA method have to do with managing the improvement initiative according to the detailed improvement plan and testing solutions in pilot projects before institutionalising them.

The references used for analysing QIP fail to emphasise commitment for improvement at any level. Training is ignored as well, as is also the management of SPI initiatives according to the improvement plan. Quite surprisingly, there is neither any explicit reference to practical support for software development projects to be found nor are the improvement goals linked with the business goals. Thus it may be concluded that QIP makes a distinction between organisational and project level improvement activities. Even though QIP has been broadly referred to as the major underlying method for SPI (Ruhe 2001), a detailed description of its functionality is lacking. Also, it remains unclear what the possible scenarios are for spinning the project and the organisational cycles in practice. The strengths of QIP lie in its tailorability, in the plan and check phases, and in the focus on continuity.

The IDEAL and ISO 15504 Part 7 improvement methods both did quite well in the CSF comparison, their results differing only slightly. IDEAL advocates SW-CMM assessment, while ISO 15504 Part 7 has been developed to be used together with ISO 15504 assessment; accordingly, the two methods do not support different SPI approaches. In both methods, commitment is regarded as important but is not clearly sought at engineer level, nor are new solutions developed with engineers but rather by separate, dedicated groups, e.g., by the Software Process Engineering Group (SPEG) in the IDEAL method. Both build improvement actions strictly on process assessment results, though ISO 15504 Part 7 considers cultural aspects too. The continuous improvement model of ISO 15504 allows flexibility in the improvement targets, but the staged improvement model of SW-CMM requires all processes to be on a certain level, thus strictly channelling the needed improvement. Certain SW-CMM maturity levels may be required by some customers, which is why these maturity levels may constitute a real need for organisations. However, this is not quite in line with the spirit of this success factor. Both IDEAL and ISO 15504 generate detailed improvement plans. Contrary to ISO 15504, IDEAL considers piloting important, while both methods lay emphasis on the support for software development projects during the improvement initiative. IDEAL recommends using metrics for managing the improvement initiative, whereas ISO 15504 Part 7 does not pay any attention to this. Both also acknowledge the importance of sustaining the improvement actions with feedback.

4.1.2 Software Process Quality

In the second category, software process assessment methods and software process standards are evaluated. The assessment methods provide guidance on how to determine process capability using a reference process model assessment method is build on where as the software process standards put forward an example of how software development should be organised.

Software process assessment methods such as SW-CMM, Bootstrap, or ISO 15504 Part 2 assess software development processes, comparing them with the selected reference process model. Even though appraisals do not perform too well in CSF evaluation, their strength is to be found in the capability to provide quick overview of the software development processes, thus providing starting point for improvement planning and a measurable baseline against which the effects of improvement actions can later be compared. For the Software process assessment methods questions like "How to organise the improvement" or "how to proceed after an assessment" have not been major issues to address. In the case of SW-CMM and ISO 15504, this lack is remedied by the SPI management models discussed above (Ideal and ISO 15504 Part 7). In their use, CSF Comparison SW-CMM, Bootstrap and ISO 15504 Part 2 are very similar; they focus on clarifying the current status of software development. Bootstrap as the only method fathoms out business goals first, on the basis of which the assessment scope is then planned. Moreover, there are some deviations between the models regarding how assessment itself is performed and how results are calculated and presented, but these differences are not visible in the context of the CSF Evaluation.

Software process standards, such as ISO 9000-3, provide a model of how software engineering should be organised. Still, from the SPI point of view, the ISO 9000-3 standard stands for a static software engineering process model; there is little if any guidance on how proposed processes should be established. Since the CSF criteria address the dynamic aspects of SPI, none of the success factors is fulfilled by ISO 9000-3.

SWEBOK trial version 1.00 is a collection of scientific type of papers arranged around software engineering topic areas. They define the terminology, concepts and approaches for knowledge areas and a great number of references to related research. SWEBOK is not yet a standard, but it shall come an IEEE standard after the project is finalised. The CSF criteria do not recognise any CSFs to be clearly addressed by SWEBOK either.

4.1.3 Measurement Methods

In the third class, four measurement methods are evaluated. Measurement methods are concerned with describing measurement practices and giving guidance on how to establish, collect and utilise metrics to support software development and process improvement.

GQM is a flexible measurement method that ties people from different organisational levels to measurement activities. It operates especially well on various organisational levels, which is why thus metrics are defined and interpreted by managers and engineers working together. While the method does not explicitly seek commitment, at the software engineer level commitment can be implicitly established by developing metrics together. Another shortcoming that can be substituted for can be found in the fact that, even though the GQM method does not incorporate training as part of the method, this can be overcome by applying the "learning by doing" principle during GQM feedback sessions, in which measurement results are evaluated and interpreted by project personnel. Nonetheless, the importance of training and commitment is not explicitly addressed. GQM develops metrics on a case-by-case basis, which allows cultural differences to be properly considered. GQM can be used for clarifying and understanding the current status of any process. GQM operates at the project level and the interface to higher organisational level activities is weak. Accordingly, no link between measurement goals and business goals are established. In addition to measurement goal definitions, a detailed measurement plan is developed to support measurement activities. For this reason, the detailed improvement plan success factor can be regarded as fulfilled. Support is provided to engineers in the form of GQM feedback sessions. GQM does not explicitly propose any practices or metrics for process follow-up, nor does it try to ensure sustainability of improvement or measurement actions.

Statistical Process Control (SPC) focuses on controlling the process and keeping it within defined upper and lower boundaries. It utilises the data seeking similarities, differences and explanations of projects using statistical means and aiming for stabilising the process and making it capable. SPC addresses only few of the success factors found critical for SPI, and it uses existing data for controlling the process. Improvement actions are developed on the basis of the process-related analysis. Thus it may be assumed that the solutions are developed on a case-by-case basis. SPC addresses business goals, and the issues for developing measurements are derived from those goals. Although the emphasis of the method is clearly on statistical means, the method also incorporates some elements of improvement.

Practical Software Measurement (PSM) is a method guiding project level measurements using such project issues as risks, problems and concerns as initiators. However, operating strongly at the project level, PSM does not aim to create any links from measurement to higher-level goals. Commitment for measurement activities is not clearly sought, either, though co-operation in measurement activities at project level may implicitly commit project personnel to measurement. Measurement activities are planned in detail, training needs are well addressed, and the project is also supported during metric collection and analysis. In spite of a detailed measurement planning, neither current status of measurement nor software development processes is considered to support this planning. PSM also incorporates a separate activity for improving the measurement activity itself by evaluating results and performance measures. Among the measurement methods PSM provides the best coverage of CSFs.

Whereas GQM and PSM are project-level measurement methods, Balanced Scorecard (BSC) embodies the top-down organisational view on measurements. BSC is primarily a management method designed for incorporating all organisation stakeholders under the same management schema. Via its steps the method seeks commitment at all levels and makes an attempt to link business goals to measurement. The general goal is to join everyone under one measurement-guided management system. One of the main features and the focus of BSC are to be found in the operationalisation and balancing of business goals in four areas and developing specific metrics for them. BSC not being a software-specific method, however, little attention is paid by to the development of lower level metrics, e.g., software development metrics. BSC addresses quite a few CSFs, including a unique philosophy concerning how to use measurements for guiding the overall actions of an organisation. In conclusion, BSC still remains a high-level measurement method with little practical support for software development or SPI.

4.1.4 Product Quality

The goal of SPI is often to raise the quality of software. Product quality standards characterise software product quality by naming and describing several aspects relevant to quality. For this reason, product quality standards are considered here as well.

In the product quality class, there are two standards to evaluate: the ISO standard 9126 for software quality characteristics and metrics, and the IEEE standard 1061 for software quality and metrics methodology. From the SPI point of view, neither of these contains any reference quality model to pursue or any improvement activities. Furthermore, due to their static nature they do not fulfil any CSFs.

4.1.5 Knowledge Management

Experience Factory (EF) is concerned with the reuse of the software engineering experiences, knowledge and results of SPI. The lack of detailed descriptions of the EF method undermines the evaluation, although there are quite a few scientific publications available on EF. The EF method can be seen as an enhancement to QIP, as it is built on QIP processes. Similarly to QIP, commitment to improvement activities or training have not gained attention, nor are the improvement activities of EF guided by business needs. Furthermore, it is not clear how the improvement goals for EF are set, planned, tracked or evaluated. In principle, new solutions are developed based on what software engineers provide for EF in the form of project models and data. Due to the bottom-up approach, EF should ensure that project, organisation or country specific characteristics are considered. Practical support for software development projects is provided by the support organisation of EF. In EF the improvement is seen as continuous learning activity, which is sustained through the EF steps.

4.2 Conclusions

The above analysis brings out that none of the methods covers all the critical success factors defined for SPI. *Table 9* presents the results of this evaluation. A checked box "✓" means that the specific success factor is clearly addressed and highlighted by the method, while the blank denotes the feature is not addressed at all, or addressed only weakly. Still, prudence is called for when interpreting the results, and it should be borne in mind that the methods approach improvement from different angles, and that the existing references vary from detailed handbooks to scientific articles.

For their capacity of considering critical success factors, PDCA turned out to be the best method covering a total of 13 out of 15 defined CSFs. The purpose of CSF criteria is to detect issues and activities considered vital for the success of a process improvement effort. Judging by the evaluation results, it can be concluded that process quality appraisals and both process and product quality standards, in particular, fail to fulfil the criteria set for SPI. As regards standards, this is a natural outcome, as standards are static by nature, whereas CSF criteria evaluate mainly the dynamic aspects of SPI. Yet, it should be noted that process quality appraisals may be referred to as improvement methods. However, in the CSF evaluation process, quality appraisals only weakly address issues found critical for successful software process improvement. Furthermore, when examining the results in *Table 9*, the question arises if it is possible to cover all the CSFs influencing the success of improvement operations using a wise combination of methods? The questions how to combine these methods successfully and if other elements are needed will be discussed in forthcoming chapters.

Table 9. Evaluation of related research.

	SPI Management				Process Quality					Measurement				Product quality		KM
					Appraisals			Standards								
Critical Success Factor Questions (Does a method address following)	PDCA	QIP	Ideal	ISO 15504 -P7	SW-CMM	Bootstrap	ISO 15504 -P2	ISO 9000-3	SWEBOOK	GQM	SPC	PSM	BSC	ISO9126	IEEE Std 1061	EF
Improvement Management																
1. Does the method support different SPI approaches?	✓	✓								✓		✓	✓			✓
2. Does the method support participation of all affected parties?	✓									✓		✓	✓			
3. Does the method support co-operation with software engineers?		✓								✓		✓				✓
4. Does the method support planning and carrying out training as a part of the initiative?	✓		✓	✓						✓		✓				
Commitment																
5. Does the method support commitment of top managers?	✓		✓	✓	✓	✓	✓						✓			
6. Does the method support commitment of middle managers?	✓		✓	✓	✓	✓	✓						✓			
7. Does the method support commitment of engineers?	✓															
Cultural Aspect																
8. Does the method support that improved solutions are developed on a case-by-case basis?	✓	✓		✓						✓	✓	✓	✓			✓
Plan																
9. Does the method support that the current status of processes is clarified?	✓	✓	✓	✓	✓	✓	✓									
10. Does the method support that the link between business and improvement goals is established?	✓		✓	✓		✓					✓		✓			
11. Does the method support that improvement goals are measurable?	✓	✓			✓	✓	✓			✓	✓	✓	✓			
12. Does the method support that an improvement plan is generated?			✓	✓						✓		✓	✓			
Do																
13. Does the method support developed solutions are tested in a pilot project?		✓	✓							✓		✓				✓
Check																
14. Does the method support using metrics in monitoring improvement actions and results?	✓	✓	✓									✓				
Act																
15. Does the method support sustainability of an improvement initiative?	✓	✓	✓	✓								✓				✓

5. Towards an Integrated SPI Approach

5.1 Background

The development of the first comprehensive SPI approach originates as a part of this research as early as the 1990's. At that same time, the development and use of software assessment methods such as SW-CMM (Paulk et al. 1994, Paulk et al. 1993) and Bootstrap (Kuvaja et al. 1994) was in the focus of interest of numerous researchers. Simultaneously, the GQM measurement method (Basili et al. 1994a) was maturing and taken in industrial use with the first promising results (Barnard and Price 1994, Oivo 1994). It was also acknowledged that GQM alone could not adequately define a measurement program, but rather needed to be linked with a more general framework and with organisational goals (Card 1993). The PDCA improvement cycle (Deming 1986) was refined into the form of the QIP paradigm (Basili et al. 1994b), proposing general phases for an improvement program. The research efforts were focused on developing disconnected approaches, yet all of them aiming at improving the quality of software. The stated research question was how improvement should be organised and whether existing approaches could be united, and if, how should that be realised in practice.

Deming's PDCA improvement cycle stresses the continuation of actions and the use of data, both factors considered as important. Data is needed for taking rational decisions, e.g., in selecting the improvement target, following the progress of actions or evaluating the achieved results. Deming's cycle was selected also due to its straightforward but also functional and well-defined phases. Moreover, the PDCA cycle provides best coverage for the CSFs. This bedrock formed by the PDCA cycle, emphasising teamwork and collaboration software process analysis, was complemented by modelling, measurement, and improvement techniques so as to be able to satisfy the SPI success factors.

Papers I and II describe and evaluate the results of this work. The resulting approach is called Pr^{2}imer, which stands for "Practical Process Improvement for Embedded Real-Time Software". In the following sections, the main features and functionality of Pr^{2}imer are recapitulated and assessed by using CSF criteria. CSF-n refers to an answer to the CSF question presented in Appendix A.

5.2 The Improvement Process

5.2.1 Analysis of Software Process Current Status

The improvement work starts with analysing the current situation of software development processes, methods and tools. The purpose of software process analysis is to evaluate and describe current software development practices (CSF-9), to identify the most serious problems, and also to define objectives for improvement (CSF-11).

The Pr^{2}imer method recognises two forms for analysing the software process: quantitative and qualitative. Examples of the quantitative approach are provided by the use of software process maturity models such as Bootstrap (Kuvaja et al. 1994), SW-CMM (Paulk et al. 1994, Paulk et al. 1993), and the use of measurement data. Qualitative analysis methods, again, generally employ a question scheme for modelling the actual process. The qualitative analysis approach employed in Pr^{2}imer has been developed in the AMES project (Application Management Environments and Support) within the Esprit-3 project. The Pr^{2}imer method also allows using international software quality standards such as ISO 9000-3, and even follow-up of software development can be applied.

Pr^{2}imer emphasises the necessity of selecting the analysis technique or techniques according to the specific needs of the subject organisation or process. In process analysis, it is important to model the actual process in order to be able to develop improvement solutions that will best fit the target organisation or project (CSF-8). Therefore, Pr^{2}imer recognises the descriptive process modelling approach as a vital element for SPI. The best combination of methods for current state analysis is to use both quantitative and qualitative analysis techniques. Quantitative methods will help detect possible areas for improvement and provide a baseline for the SPI actions, while qualitative approaches provide guidance for improvement planning.

The results of this phase include understanding the current status of software processes, which is captured in the form of process models, maturity information, or quality models developed using measurement data. The identified problems are described and prioritised together with project personnel and managers (CSF-2, CSF-3) and used as basis for improvement goal selection.

The improvement techniques applied include reviewing and analysing quality manuals, measurement databases, project plans and other relevant project results before conducting interviews. A successful approach to problem prioritisation has been found in brain storming and in other group work techniques carried out together with persons representing various areas of software development (CSF-2, CSF-3).

5.2.2 Definition of Target State

The improved process which is expected to support achieving selected improvement goals is described in the form of prescriptive process model. The aim is to produce practical descriptions to be taken in use by a pilot project (CSF-13). Thus, a descriptive process model can be used for proposing templates, checklists or other practical guidelines for a pilot project. The format of the prescriptive process model should fit the subject organisation; for instance, if an organisation or projects are accustomed to using a specific modelling language or specific tools, those should be preferred as an option (CSF-8), while also considering any other methods and tools required by the improvement endeavour.

The improvement goals selected during the current state analysis should allow measurable goals to be defined for the SPI initiative. In Pr^{2}imer, the success and follow-up of process improvement have been implemented using the GQM measurement method (Basili et al. 1994a). The improvement goals defined earlier are translated into measurement goals in a GQM plan, aiming at measuring the achieved progress in adapting new practices and in improvement results.

The main results of the second Pr^{2}imer phase include definitions of improved practices presented in shape of process models, templates, guidelines, or other forms of instructions. Furthermore, the measurements are planned for evaluating the achieved results, but also for supporting the improvement management in the course of the initiative (CSF-14). The applied techniques comprise, once again, teamwork including interviews, brainstorming, and other creative techniques.

Process standards and reference models may also be consulted when developing better practices. Here, the Pr^{2}imer considers it important that people representing different roles in the organisation take part in planning, i.e., not only possible quality professionals but also other knowledgeable software engineering experts (CSF-2). It is of great advantage if the pilot project is already known in this phase, which will allow improvement to be flexibly tailored to the needs of the particular project (CSF-8). Pr^{2}imer uses the measurement goal definition template provided by the GQM method for structuring the measurement goals, while also employing an abstraction sheet (Gresse et al. 1995) to find the right metrics.

5.2.3 Planning of Development Measures

The third phase of Pr^{2}imer is devoted to detailed planning, which is to be completed before piloting can take place. As it is generally acknowledged that SPI involves high risks and complicated effort, the planning and tracking of the improvement process needs to be carried out with extreme care so as to lower the risk as much as possible. To facilitate the planning process of improvement implementation, the pilot project will have to be known at the time of planning, which will allow the improvement actions to be well timed and adjusted to the schedule and needs of the development project itself.

The planning phase produces precise plans describing how to proceed in practice from the current situation towards the desired process state. The improvement plan defines the responsibilities, tasks, schedule, and checking points to follow along the improvement progress, while also including planned training sessions (CSF-12, CSF-4). The planning phase involves detailed planning for measurement activities as well (CSF-14). A measurement plan is constructed to support the implementation of improvements in a pilot project and also to evaluate the success. The GQM method is used for establishing a measurement plan describing how the metrics defined are to be implemented. The measurement plan further proposes expedient procedures for metrics collection by pointing out by whom, when, and how measurements shall be implemented.

The third Pr^{2}imer phase produces plans for improvement implementation and measurements customised to the use of the planned pilot project. The preparation of these plans is carried out by SPI experts with the support of pilot project representatives (CSF-8, CSF-3).

5.2.4 Piloting and Commissioning

The fourth Pr^{2}imer phase consists of two parts: piloting and commissioning. In the piloting phase, the revised software development process and possible methods and tools are tested in a pilot project or projects. This is done to lower the risks involved in adopting new practices, and also to adjust and fine-tune the practices to best fit for the organisation in question. Any good piloting results can be used as a "sales argument" when launching the new practices on a large scale.

The progress of piloting is tracked on a regular basis and the improvements are measured according to the measurement plan. The improvement plan, prescriptive process models, templates etc. are changed if necessary on the basis of measurement results or any other feedback obtained from the pilot project. The most important purpose of piloting is to field test the practices before making the decision of adopting them for large-scale use. After piloting, the success of the initiative is evaluated using the collected measurement data. If piloting indicates improvement and the general feedback from the project is positive as well, the remodelled practices can be institutionalised within the organisation.

Pr^{2}imer recommends writing an SPI evaluation report for later use. This post mortem evaluation should consider the SPI initiative as a whole and describe the actions taken and results achieved while also pointing out any further improvement areas detected. The aim is to gather general lessons learnt to be subsequently used by forthcoming SPI initiatives.

Piloting takes up the most time of all phases, depending largely on the pilot project schedule. This phase produces measurement data, analysis results and an overall SPI evaluation report. In this phase, the project has the most active role of all actors involved, SPI experts attending the measurement analysis sessions and providing support when necessary (CSF-4).

5.2.5 Summary of the Process

Figure 18 summarises the Pr^{2}imer phases. While the phases and activities may run partially parallel, Pr^{2}imer proposes the main path to proceed. The activities in the different phases need to be revisited as needed, as is made explicit in the piloting phase. It is thus acknowledged that plans are rarely perfect at first. Pr^{2}imer identifies and links the important elements of SPI by presenting a basic improvement strategy embodying process analysis, goal setting, process modelling and measurement. The emphasis on co-operation and teamwork is also integrated in the concept.

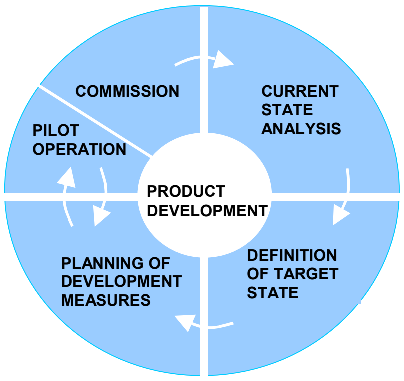

Figure 18. Pr^{2}imer improvement method.

Figure 19 illustrates the main results and phases of Pr^{2}imer. The triggering impulse for improvement may come from inside a company, e.g., as a result of the piloting and commissioning phase, or it may come from outside, e.g., in form of a customer with ISO standardisation requirements. Sometimes software development has to comply with some standard; new software development strategies may also require changes in the software development process. The change or improvement of software development processes may be triggered by a whole range of different catalysts.

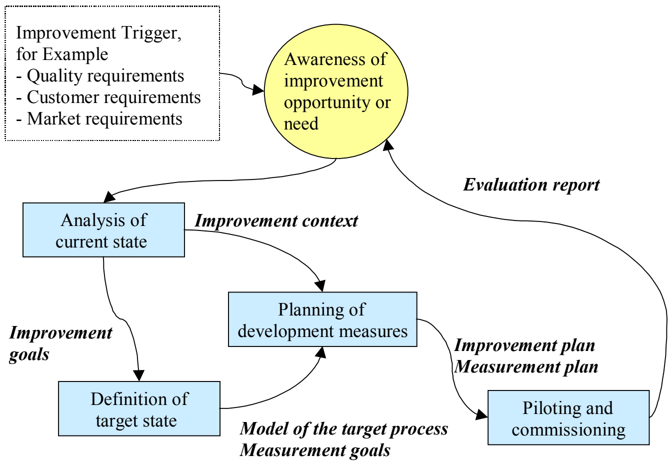
Improvement Trigger, for Example
- Quality requirements
- Customer requirements
- Market requirements
Awareness of improvement opportunity or need
Evaluation report
Analysis of current state
Improvement context
Planning of development measures
Improvement goals
Improvement plan
Measurement plan
Definition of target state
Piloting and commissioning
Model of the target process
Measurement goals

Figure 19. The functionality and results of Pr^{2}imer.

5.3 Evaluation

By 1998, Pr^{2}imer had been applied to over 20 industrial SPI cases, which provided a sound basis for canvassing customer feedback. The survey approached the issue from various angles provided by past industrial SPI initiatives, using a total of 72 questions to clarify various aspects, such as customer expectations, benefits achieved, and the applicability of Pr^{2}imer. In total, 10 answers were retrieved. The construction and process of the survey, including detailed results, are reported in (Tanner 2000).

One of the questions focused on the effects of SPI actions. Half of the respondents pointed out several implemented process changes; others stated that changes were under way. According to one respondent, the improvement had happened only on a mental level. It also turned out that although the current state analysis had opened the eyes of managers, this had not led to any concrete improvement actions. The biggest problems faced by SPI in the various organisations were the lack of SPI resources and changes in project staff and/or organisation. In a few cases, these resulted in a suspension of the improvement project.

The question focusing on respondent satisfaction (“Estimate how well the goals set have been reached?”) indicated that, when using the scale 1 to 5, the average satisfaction was 3.14. Unfortunately, no quantitative data of SPI effects could be retrieved via the survey.

“The analysis of current state” was considered the most mature phase of Pr^{2}imer by 80% of respondents, while an equal amount of replies named GQM and measurement planning as requiring the most clarification and fine-tuning within the method. Based on the survey, it can thus be concluded that the improvement areas of the Pr^{2}imer method are to be found in the clarity of measurement and in its capacity to bind improvement activities to organisation level activities.

In *Table 10*, Pr^{2}imer is evaluated using the developed CSF criteria.

Table 10. Pr^{2}imer method evaluation using CSFs.

CSF evaluation	The Pr^{2}imer method
Improvement Management	
1. Does the method support different SPI approaches?	Yes. Pr^{2}imer proposes improvement phases and activities and suggests methods that can be used during the course of work. Improvement strategy does not build on any software reference model but on the needs of the target organisation.
2. Does the method support participation of all affected parties?	Yes. This is emphasised throughout the method. Improvement goals should be selected and the improvement planned together with the people who will be affected by the improvement.
3. Does the method support co-operation with software engineers?	Yes. Pr^{2}imer builds on co-operation and teamwork with software engineers in all phases.
4. Does the method support planning and carrying out training as a part of the initiative?	Yes. The method promotes clarifying training needs and planning of training. Furthermore, practical support is given, e.g., in form of GQM feedback sessions.
Commitment	
5. Does the method support commitment of top managers?	No. The commitment of top and middle managers is not directly addressed on method description level. The commitment is assumed come about through the agreement on improvement work.
6. Does the method support commitment of middle managers?	No. The commitment of middle managers is not directly addressed.

7.	Does the method support commitment of engineers?	No. The commitment issue is not explicitly addressed. However, the involvement in selecting improvement goals and in the planning of improvement actions in co-operation with managers increases the engineers' level of commitment.
Cultural Issues		
8.	Does the method support developing improved solutions on a case-by-case basis?	Yes. The analysis findings and the needs of the organisation or project guide the improvement goal selection and improvement planning.
Plan		
9.	Does the method support clarifying the current status of processes?	Yes. In addition to the suggested use of process assessment and measurement Pr^{2}imer also supports the development of descriptive process models.
10.	Does the method support establishing a link between business and improvement goals?	No. This is not particularly stressed in Pr^{2}imer.
11.	Does the method support measurable improvement goals?	Yes. The current state of processes is analysed using diverse methods. The improvement goals are drawn out together with software engineers and managers based on analysis results.
12.	Does the method support the generation of an improvement plan?	Yes. The improvement plan is one of the two main results of the third Pr^{2}imer phase.
Do		
13.	Does the method support developed solutions are tested in a pilot project?	Yes. The forth phase of Pr^{2}imer contains a specific piloting period before the institutionalising phase.
Check		
14.	Does the method support using metrics in monitoring improvement actions and results	Yes. Metrics to follow the improvement progress are developed and analysed in regular feedback sessions. The results of improvement actions are analysed in the last feedback session.
Act		
15.	Does the method support the sustainability of an improvement initiative?	No. Though addressed by Pr^{2}imer, this feature is not explicitly stressed.

When studying the overall results of the CSF criteria evaluation, it can be concluded that although Pr^{2}imer already covers a great number of SPI-specific CSFs, there is still room for improvement. For instance, commitment could be addressed more clearly, especially at the top management level, and the link to business goals could be made more explicit for the planning process. Although measurement was already included in Pr^{2}imer, in form of GQM and measurement planning, the need to improve and clarify measurement activities was raised by the survey respondents. Otherwise, the results of the Pr^{2}imer customer survey and CSF criteria evaluation lead to an identical interpretation of improvement needs.

5.4 Summary

The research within the first engineering research cycle resulted in a proposal for an overall structure of the SPI initiative. The result broke the improvement initiative up into four phases:

1. Analysis of current state,
2. Definition of target state,
3. Planning of development measures, and
4. Piloting and commissioning.

Pr^{2}imer, the first integrated SPI method, states the desired outcome and proposes several techniques to be applied, while leaving the final decision of techniques up to the representatives of the case organisation. Pr^{2}imer further proposes how SPI should be managed by defining SPI as a project requiring to be planned as any project by clarifying goals, tasks, responsibilities, follow-up mechanisms and schedule. Moreover, the need of employing measurements and metrics is explicitly addressed and emphasised by the method.

The evaluation of Pr^{2}imer by means of the CSF criteria showed that most of the critical SPI success factors were adequately addressed by Pr^{2}imer. Pr^{2}imer performs particularly well on the project level. The weaknesses of Pr^{2}imer concerning the CSF criteria had to do with promoting commitment, expanding SPI activities to organisational level, and linking them with business goals. The customer survey also indicated that the Pr^{2}imer measurement activities require further attention and improvement. Thus, the risk of not gaining the full profit of SPI activities still exist.

6. Enhanced Role of Measurements in SPI

On the basis of the customer survey results, measurement was selected as a development area for the second engineering research cycle. The other improvement candidates were achievement of commitment and business goals related improvement goals. Also, the sustainability of improvement program presented itself as a further development issue. Measurement was selected due to it having been pinpointed by customers and due to its importance in providing a means of following, guiding, and assessing SPI results, as acknowledged by several researchers (Basili & Rombach 1988, Pfleeger & Rombach 1994, Briand et al. 1997).

Measurement is a technique that supports the management, understanding, and prediction of software development processes. In the context of SPI measurement plays many roles as well. Measurement data can be used to set improvement goals, but it can also be used to evaluate software quality and process performance, to monitor actions once improvements are being implemented, and finally to evaluate the success of the improvement.

6.1 Background

Measurement has been one of the major software research themes since the end of the 1970's, when McCabe (1976) and Halstead (1977) presented their work on software metrics. Several conferences, for example the yearly Software Metric symposium organised since 1993, and journals such as IEEE Software March 1990, July 1994 or March/April 1997 have been dedicated to pondering when, how and what to measure.

Grady (1992) divides measurement activities into tactical and strategic. The former comprises measurement activities that support project management and the latter activities that support process improvement. Basili (1993) expanded the use of GQM from tactical applications to strategic applications by showing how GQM can be applied within the Quality Improvement Paradigm (QIP) in the goal setting step. This means that the GQM measurement program could be used to characterise a current project, its environment, and, consequently, used in goal setting. However, GQM was not yet explicitly linked to other SPI activities.

From the beginning of the 1990's numerous researchers in the area of measurement support the hypothesis that measurable process improvement is both possible and desired (Grady 1992, Möller & Paulish 1993, Pfleeger & Rombach 1994, Fenton & Pfleeger 1999, van Latum et al. 1998, van Solingen & Berghout 1999, Kitchenham 2000). Despite this consensus, it has not been a straightforward success in practice, neither in tactical measurement applications nor in strategic applications for understanding and applying measurements in the course of an SPI initiative (Pfleeger et al. 1997, Rifkin 2001, Hall et. all 2001). Burgess (1995) recapitulates the papers of the "Fifth International Conference on Applications of Software Measurement" and notes how many are quick to point out the downside of measurement. Mashiko & Basili (1997) describe a study of four software development projects, which were used to build descriptive models of software process, defects, and cost in order to understand the factors influencing SPI. After thorough measurement definition, collection and analysis, the final conclusions are guarded with the words, "In characterizing the projects, we found some patterns…" and "We may have gained some new insight about the cost of defects…" (Mashiko & Basili 1997, p. 31). The results reflect both the laboriousness and some difficulty of using measurement data, especially as an initiator for SPI.

In the second engineering research cycle the strategic use of measurements in the SPI context was focused upon. The research concern was further narrowed to defining how to improve utilisation of measurements.

Paper III details how measurement activities in the SPI context can be expanded by presenting two industrial case studies and illustrating how the GQM process is explicitly bound to the Pr^{2}imer improvement phases. Furthermore, paper III discusses the tool support to ease measurement. As well, Paper IV illustrates in relation to the GQM process how measurement can be automated to lessen the extra bother measurement causes for software projects.

6.2 The Measurement Strategy

Measurement provides a tool for SPI that can be used in various ways, for example analyses of large defect database may point out places for improvement. It is proposed as well that a full measurement program can be initiated to characterise the project and organisational environment (Basili 1993, Mashiko & Basili 1997). Both of these characterizations are possible, but unfortunately they have some deficiencies. For example, the result of defect database analysis can be only as good as the quality and extent of data entered into the database. Paper III illustrates a case where coding defect was claimed to be introduced in the requirements definition phase. In this case it was obvious that there was a mistake in entering the data, and thus validity of data was a problem. Measurement databases contain flaws that are never detected due to their nature or lack of analysis by the engineers and managers who produced the data. To start SPI by establishing a measurement program takes up a lot of time, including setting up a measurement program, defining metrics, collecting measurement data and analysing it. In addition, becoming aware of general difficulties that measurement programs face in industry this is not the easiest way to start. Furthermore, many organisations are not mature enough to cope with measurements to a great extent. For example, SW-CMM recognises measurement activities only after maturity level three.

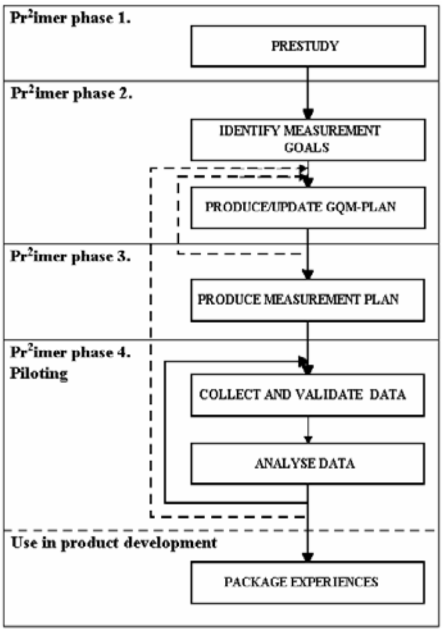

Figure 20. Primer and GQM processes (Paper III).

Pr^{2}imer recognises measurement as important part of the improvement process, but to start (Phase I), the model suggests using process assessment and/or qualitative analysis methods. From Phase I onwards Pr^{2}imer binds the GQM measurement activities tightly to the improvement process. How measurement activities are embedded in the Pr^{2}imer process is illustrated in *Figure 20.*

Two initiators exist to define measurement goals. Firstly, measurement goals are defined based on selected improvement goals to evaluate if desired improvement took place. As a simplified example, if the improvement goal is "*Improve customer satisfaction by reducing after delivery defects by 50%"* the related GQM measurement goal could be *"*Analyse *customer feedback* for the purpose of *evaluating* with respect to *the number of defects reported* from the point of view of *customers*". Secondly, measurement goals are defined to follow the implementation of improvements during the course of the SPI project (CSF-14). In order to reach the improvement goal changes in the software development process have to be planned and implemented. To continue with the example, analysis of the current status of the process may have made evident that, in general, testing procedures are good but the unit testing procedure is poorly defined resulting in inadequate unit testing. As an improvement action, an ameliorated unit testing procedure using testing tools is defined. To follow up the use of new practices the following measurement goal may be formulated as: "Analyse *the unit testing* procedure for the purpose of *evaluation* with respect to *new practices* from the point of view of *prescriptive unit testing process model*". This measurement goal may lead to the following metrics*: the number of software modules, the number of produced/reviewed unit test plans, the number of unit test reports et*c. The purpose of this kind of metrics is to observe during the course of the software development project whether new practices are actually used. *Figure 21* illustrates the two-fold role of measurements in Pr^{2}imer.

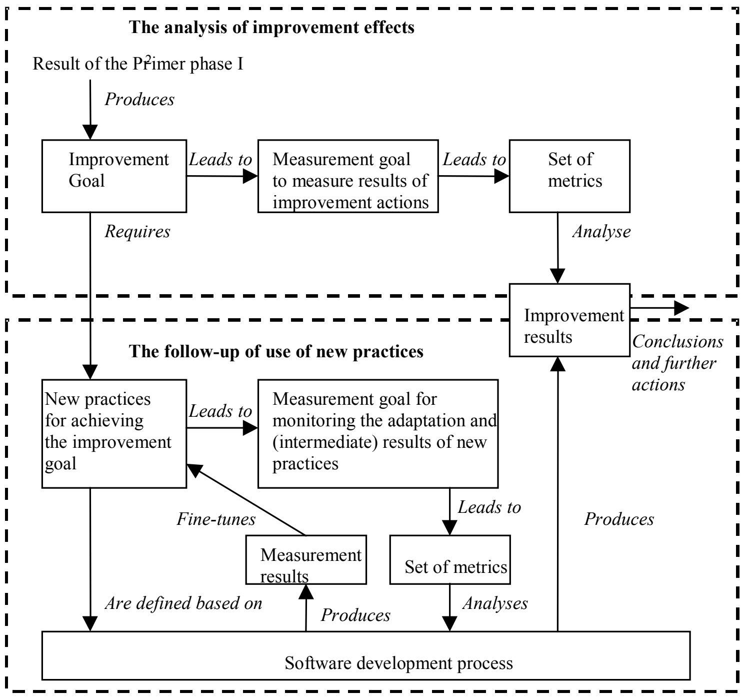
The analysis of improvement effects
Result of the Pr²imer phase I
Produces
Improvement Goal
Leads to
Measurement goal to measure results of improvement actions
Leads to
Set of metrics
Requires
Analyse
Improvement results
Conclusions and further actions
The follow-up of use of new practices
New practices for achieving the improvement goal
Leads to
Measurement goal for monitoring the adaptation and (intermediate) results of new practices
Leads to
Produces
Fine-tunes
Measurement results
Set of metrics
Are defined based on
Produces
Analyses
Software development process

Figure 21. Measurement in relation to the SPI context.

6.3 Tool Support for Measurement

Collecting and analysing measurement data can be laborious if done manually. Adequate tool support is essential in decreasing the work needed for measurement tasks, thus also helping to reduce the software engineer's resistance to measurement activities. Tool support may comprise data collection templates, measurement databases, automated data collection tools etc. The definition of proper tool support is related to the size and maturity of software development. A software development team that consists of a couple of developers requires a different approach than teams in multinational organisations with hundreds or thousands of software developers. When measurement activities are wide-ranging the automation of data collection and analysis provides opportunities to reduce measurement-related cost and offer possibilities for extensive data analysis that otherwise would require too much effort to be executed on regular basis.

To support measurement, the MetriFlame tool was developed. MetriFlame is presented in Paper III. Paper IV describes the specific features of the measurement automation process.

6.3.1 MetriFlame

MetriFlame is a measurement tool for collecting measurement data, defining metrics, calculating them and presenting the results in various ways. Typical sources for measurement data are documents and databases that are created during the course of the software development process. The procedure for collecting the data has been rendered as imperceptible as possible, thus keeping its hindrance to the software development process small. The core idea of MetriFlame is to be able to use data from different sources, process the data according to the defined GQM plan and support data analysis by providing possibilities for various presentation formats. *Figure 22* shows the MetriFlame architecture at a high level.

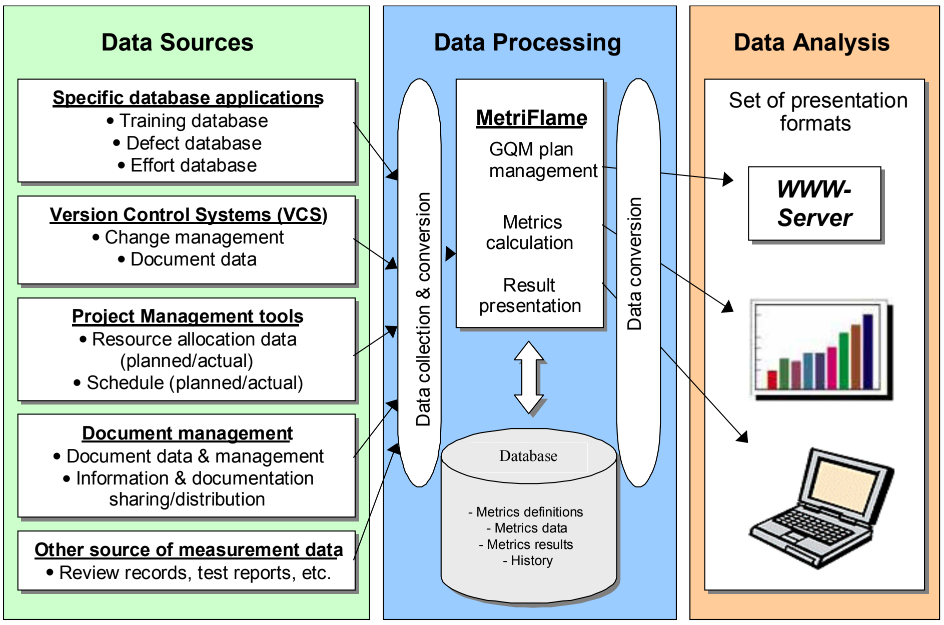

Figure 22. MetriFlame measurement environment (Paper IV).

When the development of MetriFlame started in 1996 there were no measurement tools that supported flexible data collection from various sources as defined by the user. There were several metric management tools like PC-Metric, Archimedes BugBase, Metricate etc. (Sulka-P 1994, Ronkainen 2003), but these tools were only capable of collecting a fixed set of metrics where data had to be entered manually. Furthermore, the existing commercial tools at the time did not provide any support for goal-driven measurement programmes with variable sets of metrics tailored to the needs of a particular project or organisation (CSF-8). The main functions of MetriFlame include GQM plan creation, and management of measurement data and results. MetriFlame is an example of flexible tool support for measurement. The experiences of its usage can be found in (Parviainen et al. 1997).

6.3.2 Measurement Automation

The idea of measurement automation arose from the use of MetriFlame. The idea was supported by a general awareness of the importance of automating all that can be automated. Collection and analysis according to predefined formulas as required by MetriFlame provided a natural basis for automation experimentation. Paper IV outlines the measurement automation process and describes the experiences of an industrial case study. In *Figure 23*, the measurement automation process is seen through Pr^{2}imer and GQM processes. The advantages of measurement automation are present in the pilot stage where a software development project experiments with new practices.

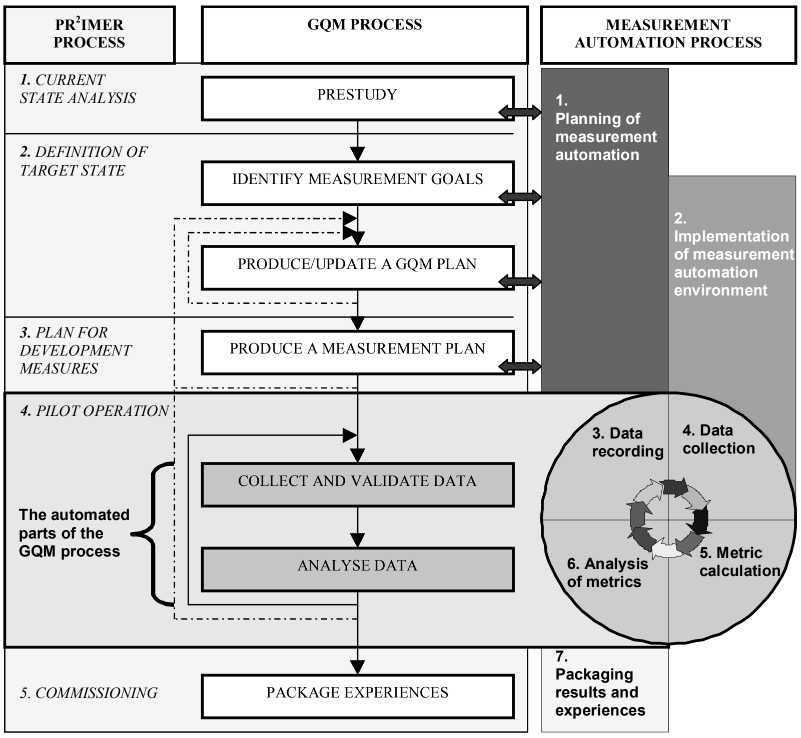
PR²IMER PROCESS
GQM PROCESS
MEASUREMENT AUTOMATION PROCESS
1. CURRENT STATE ANALYSIS
PRESTUDY
1. Planning of measurement automation
2. DEFINITION OF TARGET STATE
IDENTIFY MEASUREMENT GOALS
PRODUCE/UPDATE A GQM PLAN
2. Implementation of measurement automation environment
3. PLAN FOR DEVELOPMENT MEASURES
PRODUCE A MEASUREMENT PLAN
4. PILOT OPERATION
The automated parts of the GQM process
COLLECT AND VALIDATE DATA
ANALYSE DATA
3. Data recording
4. Data collection
5. Metric calculation
6. Analysis of metrics
5. COMMISSIONING
PACKAGE EXPERIENCES
7. Packaging results and experiences

Figure 23. Measurement automation in relation to SPI and measurement processes (Paper IV).

Measurement automation is most feasible if an organisation already has working measurement practices in place. Where this not being the case, the recommended and natural way to proceed is first to establish measurement practices and culture.

6.4 Evaluation

Table 11 accumulates the evaluation of research results after the second engineering research cycle where the measurement activities were enhanced in the context of Pr^{2}imer. No new success factors are reached but many of the already fulfilled ones are strengthened. Because of the nature of enhancement the CSF evaluation is done from the strengthened measurement practices viewpoint only.

Table 11. Evaluation of results after strengthened measurement activities.

CSF evaluation	Pr^{2}imer with strengthened measurement
Improvement Management	
1. Does the method support different SPI approaches?	Yes. though addressed already by Pr^{2}imer. The starting point for measurement definition is the software development project in question. Moreover, the measurement collection and analysis process is tailored project by project basis.
2. Does the method support participation of all affected parties?	Yes. In defining and using metrics, the GQM principles highlight the participation of all affected.
3. Does the method support co-operation with software engineers?	Yes. Software engineers participate in defining metrics and analysing the results in GQM feedback sessions.
4. Does the method support planning and carrying out training as a part of the initiative?	Yes. Measurement training is provided by GQM feedback sessions.
Commitment	
5. Does the method support commitment of top managers?	No. This is not particularly addressed.
6. Does the method support commitment of middle managers?	No. This is not particularly addressed.
7. Does the method support commitment of engineers?	No. This is not particularly addressed.
Cultural Issues	
8. Does the method support developing improved solutions on a case-by-case basis?	Yes. Software engineers and managers participate in measurement definition.
Plan	
9. Does the method support clarifying the current status of processes?	Yes. Though measurements practices do strengthen this already well addressed CSF.

10. Does the method support establishing a link between business and improvement goals?	No. This is not addressed.
11. Does the method support measurable and well understood SPI goals?	Yes, though addressed already by Pr^{2}imer.
12. Does the method support the generation of an improvement plan?	Yes. A measurement plan as part of improvement plan is generated.
Do	
13. Does the method support developed solutions are tested in a pilot project?	Yes, though addressed already by Pr^{2}imer.
Check	
14. Does the method support using metrics in monitoring improvement actions and results	Yes, further addressed with enhanced measurement activities.
Act	
15. Does the method support the sustainability of an improvement initiative?	No. This is implied but not stressed.

6.5 Summary

The use of measurement data provides a solid base for the management of improvement programmes and evaluation of the achieved results thus measurement should always be part of an improvement programme. In Pr^{2}imer, the GQM method is favoured for various reasons. It is a top-down measurement method that starts the definition of metrics from measurement goal definition. Pr^{2}imer adds an antecedent level to this, an improvement goal level that guides measurement goal definition. The co-operative nature of GQM is another reason for its utilisation. GQM develops metrics case-by-case thus taking into account the needs of software development projects and the organisation in question. Furthermore, the measurement feedback sessions as embodied by GQM have turned out to be important for data validation and right interpretation. Defining how measurements are linked to SPI and how mature measurement environments can take advantage of measurement automation complements the measurement aspect of SPI. Even though no new CSFs are clearly addressed many already fulfilled ones are further strengthened with measurement approach.

7. Product Quality Focused SPI

7.1 Background

"The underlying premise of software process management is that the quality of a software product is largely determined by the quality of the process used to develop and maintain it" (Paulk et al. 1994, p. 8). Among the SPI research and practitioners community this hypothesis is supported widely: Software quality is dependent on the quality of process, and the better the software development process is, the better the quality of the software becomes. The above-mentioned hypothesis forms the basis of this research as well; the "method development engineering research cycles" are based on this hypothesis.

In addition to "quality of the software development process" viewpoint, another important viewpoint was not yet explicitly used by any SPI method: quality from the business point of view. What kinds of quality factors are important in a product? Required, or needed, quality level differs from product to product and is context related. Flaws tolerated in word processors in an office environment may be safety-critical if occurring in hospital operation theatres' monitors. Quality factors critical for one product may be less valuable for another, but this fact was neither recognised nor explicitly addressed by any SPI method. Software quality models have been studied by many. There exist models for expressing product qualities such as McCall's quality model (McCall et al. 1977), Boehm's model (Boehm et al. 1978) or ISO 9126 (2001), but none of these static quality models are connected to any SPI approach or model. This leads to the question: How could improvement of a desired software quality factor be supported by an SPI method?

The initiator for SPI was to be changed from process quality to software product quality. One of the research questions under study in the third engineering research cycle was to find means to identify the dependencies between required product quality and influencing process or processes. Thus, business needs should become visible as SPI drivers. In method development, the following aspects were considered:

– How to link customer-driven product quality requirements to process characteristics (CSF-10);

– How to guide an organisation to focus improvement actions on those parts and characteristics of the process that are critical for achieving the required quality; and

– How further to combine and enhance the strengths of goal-oriented measurement, process assessment, product and process modelling, and how to take advantage of the “experience factory” concept.

Since how to manage and measure SPI was already known, effort was now focused on how to do it more effectively. Improvement resources should be able to be applied to targets with the best return on investment. Accepting the hypotheses that quality of product is influenced by the quality of process and that needed quality varies by product, the necessity to change the SPI initiator was discovered.

The change in the improvement viewpoint is described in Paper V, which describes how to shift the improvement focus from process quality to end-product quality. It explains the product quality driven improvement methodology Profes that supplements the Pr^{2}imer method with product quality focus. Paper VI recapitulates the status of the Profes improvement methodology and design rationale. In addition to papers IV and V the full description and functionality of the method is presented in the Profes User Manual (1999). The table of contents of the manual can be found in Appendix C. The Profes improvement methodology will be briefly recapitulated in the following sections.

7.2 Change in the Improvement Strategy

The underlying idea for the third method development research cycle was that improvement driven by specific software quality requirements (CSF-8), instead of just general process quality requirements should result in better focused and more efficient process improvement. Eventually, from the business point of view, product quality assessed by users or customers is the factor that should drive improvement decisions (CSF-10). It was also noted that time-to-market requirements and cost might set constraints on or demands for improvement activities and require balancing between them (Paper V).

Where Pr^{2}imer is built on the PDCA improvement cycle (Deming 1986) Profes operationalises the QIP paradigm (Basili et al. 1994b) by describing in detail the improvement process and enhancing it by introducing the new concept of Product-Process-Dependency (PPD) (Oivo et al. 1999). QIP was selected due to its capability of distinguishing organisational and project level processes and for its guidance in packaging and reusing lessons learnt. QIP has a close relationship with the GQM method (Oivo & Basili 1992) that had already proven useful and led to good results in SPI projects based on the Pr^{2}imer method (Paper I, Paper II, Paper III and Paper IV). The new PPD technique is developed for finding and explicitly expressing product quality and process interrelationships (Hamann et al. 1998, Birk et al. 1998). Developed PPD models are used to propose focused process changes, so that desired product quality improvement can be reached efficiently.

7.3 Upgraded Improvement Process

An overview of the phases and related steps in the Profes improvement methodology are illustrated in *Figure 24* which is accompanied by a short phase description.

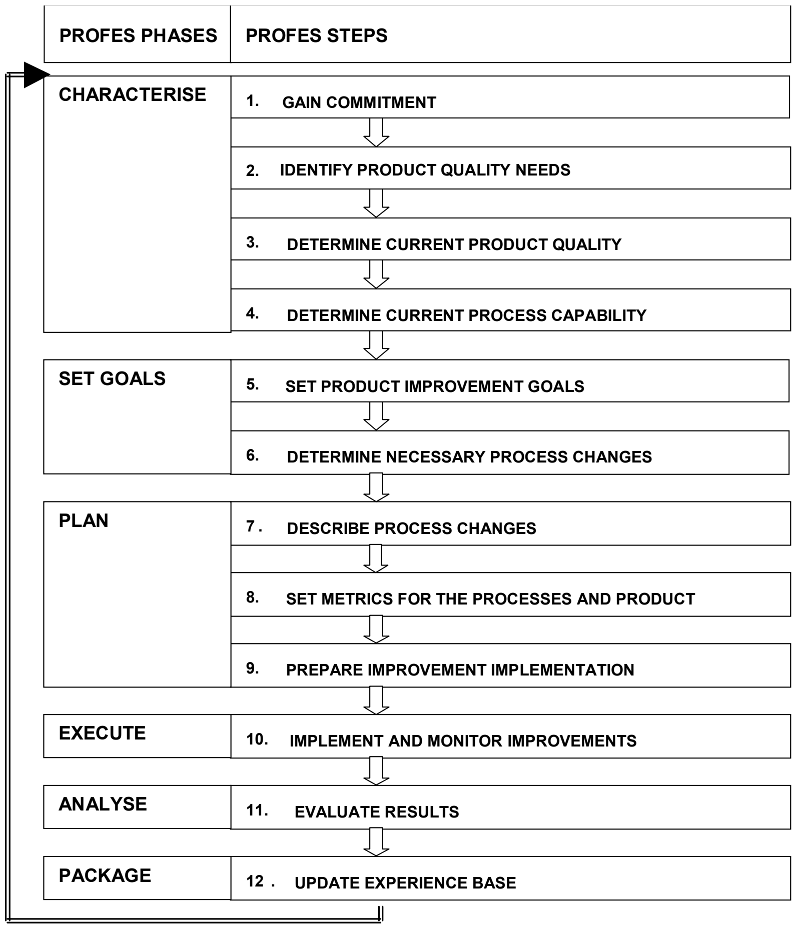
PROFES PHASES
PROFES STEPS
CHARACTERISE
1. GAIN COMMITMENT
2. IDENTIFY PRODUCT QUALITY NEEDS
3. DETERMINE CURRENT PRODUCT QUALITY
4. DETERMINE CURRENT PROCESS CAPABILITY
SET GOALS
5. SET PRODUCT IMPROVEMENT GOALS
6. DETERMINE NECESSARY PROCESS CHANGES
PLAN
7. DESCRIBE PROCESS CHANGES
8. SET METRICS FOR THE PROCESSES AND PRODUCT
9. PREPARE IMPROVEMENT IMPLEMENTATION
EXECUTE
10. IMPLEMENT AND MONITOR IMPROVEMENTS
ANALYSE
11. EVALUATE RESULTS
PACKAGE
12 . UPDATE EXPERIENCE BASE

Figure 24. Profes phases and steps (Profes User Manual 1999).

7.3.1 Characterise

In the Characterise phase, the bedrock for all forthcoming activities is laid and assured. The objective is to achieve organisational commitment to improvement, clarify product quality needs, and gather baseline data of processes and product to initiate actual planning and preparation of the improvement programme. The main activities in the Characterisation phase are:

- Establish commitment to improvement,
- Identify existing product quality improvement needs and determine current product quality, and

– Characterise and determine the process environment in which the improvement programme will be executed.

First, commitment to improvement is contrived or ascertained from the organisation to assure, or re-assure, the existence of the all-important sponsor for the improvement programme. Not only is the commitment of the organisation pinpointed but also that of all quarters involved (CSF-2) including software development team members as well (CSF-5, CSF-6, CSF-7).

When desired product quality is a starting point product improvement needs are identified first, using customer feedback, market trends, and other available sources. Here it has to be noted that the needs of customers may be conflicting and all the needs customers have regarding product quality will not take the shape of product quality goals. Product quality goals are later defined based on analysed and prioritised needs (CSF-11).

In addition to product quality needs the current product quality is analysed. Profes suggests using the ISO9126 (2001) product quality model for quality characterisation, but other quality models may be used as well (CSF-1). The current product quality is characterised for better understanding. For example, this characterisation can be based on available product measurement data, such as defect data.

Needed improvement actions can be later rationally planned only if current processes are characterised too. In order to have both an understanding of current status of software development and management processes and a quantitative basis to follow the effects of process improvement ISO 15504 (1998), a compliant process assessment method, such as Bootstrap (Kuvaja et al. 1994), is suggested to be applied (CSF-9). Here again, other process assessment methods such as CMMI (2000) may be used as well (CSF-1). Profes favours these methods because they focus, when needed, on a set of processes without having to cover them all, as for example SW-CMM requires (Paulk et al. 1993). Process maturity information is supported by descriptive process modelling, which documents how work is executed in practice. Here, the qualitative analysis approach as suggested by the Pr^{2}imer method is utilised.

7.3.2 Set Goals

In the Set Goal phase, the change in improvement strategy becomes most visible. The initiator in defining improvement goals is the improvement needs of the product. After agreeing on the product quality goals based on the needs clarified earlier, the process improvement goals are set and needed process changes are defined.

The Profes methodology instructs that the product quality goal setting has to be guided by the overall business goals of the company (CSF-10). When defining needed process improvement, the expert knowledge augmented with results of process assessment and evaluation are used as the basis. Here, PPD models that represent experimental knowledge of the processes that have significant impact on the achievement of certain product qualities are used. For instance, an improvement in requirements analysis processes can be particularly important in achieving high product reliability. In addition to general PPD models (Profes PPD Repository 1999), an organisation may maintain its own PPD model experience base. If relevant product-process dependencies are not readily available, Profes provides means to identify them, e.g., from existing measurement databases or assessment results, or with the aid of interviews or other knowledge acquisition techniques.

7.3.3 Plan

The Plan phase defines how the set improvement goals are to be reached in practice. This includes prescriptive process modelling that describes process changes needing to be implemented in order to achieve the required product quality. To be able to plan, Profes remarks that the software development project or projects in which the improvement actions shall be implemented should be known, and moreover the project members should participate in improvement planning (CSF-3). For follow-up and evaluation purposes, GQM measurement principles (Basili et al. 1994a) are applied, and a GQM measurement plan is defined with appropriate process and product metrics (CSF-14). The measurement planning includes description of the measurement process, measurement frequency, and utilised information sources. In addition to target process and measurement planning responsibilities, reporting policies are determined as well, and further supporting activities, such as training, are scheduled (CSF-4, CSF-12).

7.3.4 Execute

In the Execute phase, product quality driven process improvement actions are carried out according to the plans in the selected software projects, and the defined measurement data is collected, regularly analysed in GQM feedback sessions (CSF-4, CSF-14) and used to monitor and manage progress and goal achievement. In the course of the improvement implementation lessons learnt and other relevant experience are identified and recorded (CSF-15). When seen necessary, the corrective actions for any plans during improvement project execution should always be acknowledged.

7.3.5 Analyse

The objective of the Analyse phase is to clarify whether product quality has improved as required, and whether the process changes introduced have been effective in reaching this goal. Analysis is mainly based on the collected measurement data in GQM feedback sessions. These sessions are understood as a core learning activity that drives continuous improvement (CSF-15). After improvement activities are implemented, one potential way to analyse the effect of the improvement project is to conduct process re-assessment for those processes that were amended, which will help to understand the changes in process performance. During the analysis phase, experiences should be gathered from the areas involved in product improvement. Examples of important areas are: experiences of implemented process changes, used measurements, developed or enhanced PPD models, models for project planning, lessons learnt in general, etc.

7.3.6 Package

The Package phase translates the results of analysis into reusable objects from which future projects and improvement programmes can benefit (CSF-15). Therefore, the analysis results need to be packaged into a form that easily facilitates reuse. For instance, suggested improvements to the software development process should result in changes to the organisation's process handbook. In packaging, the Experience Factory infrastructure (Basili et al. 1994b) facilitates the understanding of efficient experience retrieval, organisational learning and knowledge reuse.

7.3.7 Summary of the Process

The Profes improvement methodology aims to create a link between the software development process and product quality. This is illustrated by the Profes improvement cycle in *Figure 2*. The link has been earlier either implicit or missed completely, e.g., when improvement has primarily focused on reaching higher maturity levels. As well, when process improvement is clearly linked to achievement of the required product quality, the effects of improvement initiatives becomes more visible.

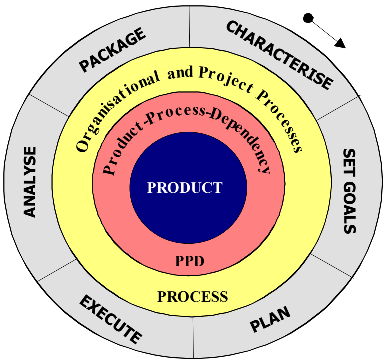

Figure 2. The QIP based Profes improvement cycle.

Profes improvement phases and main activities are summarised in *Table 12*.

Table 12. An overview of the Profes improvement phases and main activities.

Phase	Main activities
1. Characterise	Identify product quality needs and current product quality; characterise the environment of the improvement programme
2. Set Goals	Set product improvement goals, identify process improvement needs, and define improvement goals in measurable terms

3. Plan	Plan and prepare the improvement programme; select (pilot) projects to implement improvements (CSF-13)
4. Execute	Perform (pilot) projects and collect measurement data
5. Analyse	Analyse (pilot) projects and evaluate achievement of improvement goals
6. Package	Package experiences and ensure use in future projects

7.4 Evaluation

The Profes improvement methodology was developed with three industrial partners (LM Ericsson, Dräger, and Tokheim) who provided their software development environment for method development and trial use. The industrial experiences of Profes are, e.g., described by (van Solingen at al. 1999a, van Solingen et al. 1999b, van Solingen at al. 1999c, Bicego et al. 1999, Birk et al. 1998). In addition to industrial experiences and Papers IV and V, the comprehensive Profes User Manual (1999) is used here to evaluate the method as well.

Table 13. The Profes improvement methodology evaluation using CSFs.

CSF Evaluation	Product quality focused SPI
Improvement Management	
1. Does the method support different SPI approaches?	Yes. Profes utilises several improvement methods during the course of an SPI initiative and proposes alternative methods to use as well. Moreover, Profes provides ways in which an organisation can tailor the method.
2. Does the method support participation of all affected parties?	Yes. This is strongly highlighted, for example commitment and involvement of software engineers is pinpointed as an issue by the method. For example various stakeholders should be included to the work and consulted regarding quality needs.
3. Does the method support co-operation with software engineers?	Yes. Co-operation is seen as a way to commit software engineers to improvement.
4. Does the method support planning and carrying out training as a part of the initiative?	Yes. Planning of training is part of the plan phase, especially steps 9 and 10 in the Profes User Manual are concerned with training issues. The improvement initiative is managed by a Profes team showing three roles: manager, experts and support roles. This team provides support and guides actions during the course of an improvement project (Profes User Manual 1999).

Commitment		
5.	Does the method support commitment of top managers?	Yes. This is an issue of the first phase of Profes but raised later in the form of re-assuring commitment. Furthermore, using business needs, market analysis and product quality needs as the starting point for SPI binds managers to the decision process.
6.	Does the method support commitment of middle managers?	Yes. For example, in step 5 (Profes User Manual) when product quality goals are set, the achievement of management commitment is crucial.
7.	Does the method support commitment of engineers?	Yes. Step 1 Verify Commitment tries to ensure that commitment exists from everyone involved including software project members. The role of software engineers is essential for success.
Cultural Issues		
8.	Does the method support developing improved solutions on a case-by-case basis?	Yes. This is an inbuilt and emphasised improvement strategy of Profes.
Plan		
9.	Does the method support clarifying the current status of processes?	Yes. This is an important part of the Characterise phase.
10.	Does the method support establishing a link between business and improvement goals?	Yes. Product quality needs arise, inter alia, from business needs and analysis. These needs are used in setting the improvement goals.
11.	Does the method support measurable improvement goals?	Yes. Current product quality needs guide improvement goals set in co-operation with a software development project (Set goals phase).
12.	Does the method support generating an improvement plan?	Yes. This is part of the Plan phase. Profes provides, for example, a template for this plan (Profes User Manual, 1999, Appendix 2).
Do		
13.	Does the method support developed solutions are tested in a pilot project?	Yes. In the general guidance part of the Profes User Manual there is the advice to test solutions in a pilot project.
Check		
14.	Does the method support using metrics in monitoring improvement actions and results?	Yes. This is emphasised by Profes. Step 8 guides the setting of metrics for processes and product improvement; Step 10 instructs on the collection and analysis of metrics and Step 11 to evaluate the results.
Act		
15.	Does the method support the sustainability of an improvement initiative?	Yes. This is approached in several ways. Steps 11 and 12 guide evaluating results and lessons learnt and updating the Experience Base. Setting up the infrastructure for the current improvement project as well as sustaining the improvement within an organisation is guided as well.

When analysing the results of *Table 13* it can be noted that in the method level Profes addresses all factors that are understood to be critical for SPI success.

7.5 Summary

Profes enhances and packages various SPI approaches (including process and product assessment, process modelling, measurement, and experience factory) as a form of product quality focused process improvement methodology. Profes proposes 6 improvement phases and 12 related steps with well defined set activities, methods, techniques, tools, templates, work products, resources roles, and other practical instructions. As a new improvement strategy Profes understands product quality as an initiator and a starting point for an improvement programme and therefore focuses on finding and using relationships that exist between product quality characteristic and process quality. Focusing improvement efforts on those processes that will return on the investment most effectively should avoid wasting meagre resources that organisations often have to allocate for SPI. Known resource constraints including time, money, and persons, are the reasons why Profes focuses on the improvement activities rather than trying to improve all processes equally. *Figure 26* illustrates the SPI paradigm shift between the second and third engineering research cycles.

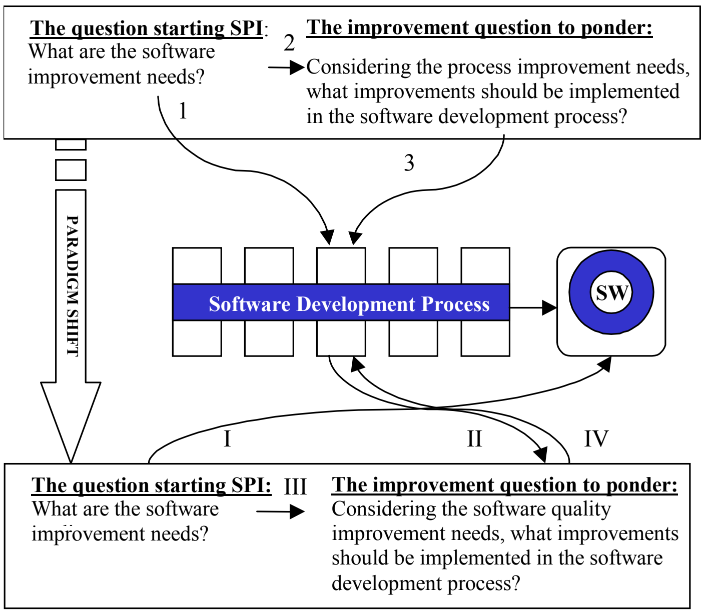
The question starting SPI:
What are the software improvement needs?
2
The improvement question to ponder:
Considering the process improvement needs, what improvements should be implemented in the software development process?
1
3
PARADIGM SHIFT
Software Development Process
SW
I
II
IV
The question starting SPI:
What are the software improvement needs?
III
The improvement question to ponder:
Considering the software quality improvement needs, what improvements should be implemented in the software development process?

Figure 26. The paradigm shift in SPI.

SPI started with process analysis to clarify what the software process improvement needs are (1). Based on this understanding of deficiencies (2) the needed changes in software development were planned and implemented (3). The new paradigm suggests that SPI should start with clarifying what the product quality needs are (I). By understanding the deficiencies of software development (II) in relation to product quality needs, the needed improvements are planned (III) and implemented (IV).

Evaluating Profes with CSF (*Table 13*) indicates it to fulfil all factors understood as important for SPI success. Profes answers also those CSFs that were not stressed by Pr^{2}imer or Pr^{2}imer enhanced with measurement focus. The new factors covered are commitment related (CSF-5, CSF-6 and CSF-7), the influence of business goals on improvement goals (CSF-10), and the sustainability of improvement initiative (CSF-15). Related to sustainability, the knowledge management issues entered SPI method development research in the third method development research cycle. Profes employs the Experience Factory approach by packaging the results in an Experience Base for further use. Despite this, it was noted that there was room for improvement in reusing lessons learnt in SPI.

8. Knowledge Management Supported SPI

8.1 Background

It is commonly agreed that software engineering is a knowledge-intensive activity (Basili et al. 2001, Henninger & Schlabach 2001, van Solingen & Berghout 2001, Rus & Lindvall 2002). Consequently, it may be argued that SPI is very knowledge-intensive activity. Besides required software engineering knowledge, people involved in SPI have to be aware of SPI models, methods, tools and techniques. SPI experts also need to know how and when to apply different techniques, what past experiences and lessons have been learnt, what the company culture is, what the business objectives are, etc. SPI is not a process that can be entirely automated and, once installed, executed as desired. Unfortunately quite to the contrary, SPI needs skilful and knowledgeable persons who understand both software and improvement engineering, for SPI is an activity that combines technical (software and improvement engineering), managerial (software and SPI management) and even human behaviour related aspects (cultural and psychological perspective) in a complex business environment. Due to these complicated interrelationships, SPI becomes an intricate task to execute and thus help from other disciplines is looked for.

In the context of software development two main streams to approach and utilise KM can be distinguished, yet both of them share the same goal: to improve software development (*Figure 27*). The first approach studies ways in which software development itself could be supported with the ideas of KM. This may mean, for example, studying in what form and ways the product requirements should be communicated to the software project team, or how best to capture and provide information about software development project needs during the course of development. For example, Kucza et al. (2001) describe experiences with improving software reuse process utilising KM. Birk & Tauz (1998) present a process and general organisational infrastructure for generalising existing still non-reusable experience statements into a form of lessons learnt repository. The emphasis is on how to transform and store existing experiences in a reusable format.

Acknowledging the knowledge intensity SPI in overall has, the other stream explores how SPI activities could be supported via KM. For instance, Pourkomeylian (2002) studied SPI from the KM point of view and concludes that the key challenge for SPI is knowledge sharing. Furthermore, various tools that support KM has been of great interest to many. KM tools comprise both enterprise level KM tools (Wei et al. 2002) and smaller specific applications to acquire and save information (Lindvall et al. 2001). Dingsøyr (2002) studied how intranet-based KM tools can support building a learning environment and examined how KM tools in general are used in organisations. The study pointed out that many KM tools organisations had acquired were later abandonment as unhelpful. Henninger & Schlabach (2001) claim that instead of static search engine based applications, a tool that alerts software engineers of relevant knowledge leads to better results. Based on five years experience building and using repositories to support software engineering at DaimlerChrysler, Schneider and Hunnius conclude: "Without a learning attitude and some appreciation for continuous process improvement, even the best repository will not make experiences 'fly' (Schneider & Hunnius 2003, p. 539). KM tools, including experience repositories or databases, are solely insufficient to improve SPI activities and to help software development, but they provide important technologies to acquire, organise, archive, search, and push information.

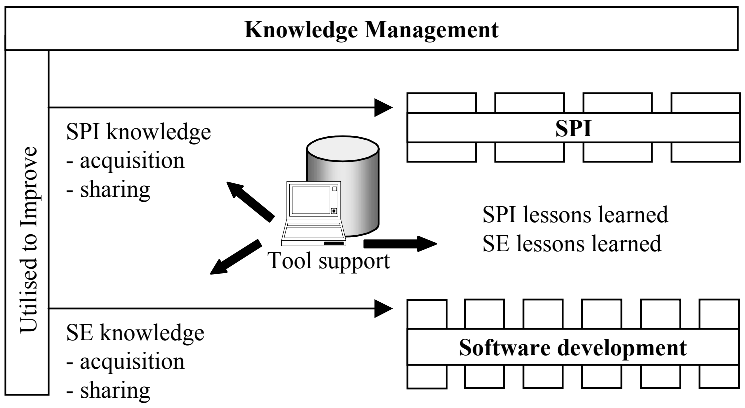

Figure 27. KM in the context of SPI and software engineering.

The KM research studying software development and improvement builds greatly on the ideas of Nonaka & Takeuchi (1995) by adapting the definitions of tacit and explicit knowledge and transformations between them. In addition, the

Experience Factory concept (Basili et. 1994b) proposing organisational infrastructure for experience collection and packaging in a form of Experience Base is researched as well. The Experience Factory developed in the early 1990's at the NASA Software Engineering Laboratory was soon also adapted to a learning-organisation concept (Basili & Seaman 2002). Improving by learning is the inbuilt idea of both Experience Factory and KM.

In the previous three method development research cycles, the one fundamental idea has been to start the improvement from a project or product perspective instead of general improvement activities over the organisation. So far this focused improvement path had turned out to be fruitful. This notification with awareness of the possibilities KM could provide to SPI initiated the forth method development cycle. During that it was studied how KM, as presented by Nonaka & Takeuchi (1995) and the software development specific Experience Factory approach (Basili et. 1994b), could be operationalised in a software development project environment in practice.

The output of this research is described in Paper VII. In the following, its main results and observations are recapitulated.

8.2 Towards a Knowledge Capturing Solution

The study was conducted in an independent business unit of a global corporation developing software-intensive electronic products (Tauriainen 1999). Pr^{2}imer (Papers I, II, III, IV) and Profes (Papers V and VI) method principles were applied to structure this KM originating improvement initiative. The organisation was looking forward to improving the software development knowledge capturing and reusing process. There had already been several attempts to improve knowledge reuse, which provided an opportunity to learn from experiences of past knowledge capturing and sharing efforts.

8.2.1 Lessons Learnt from Past Improvement Attempts

The setting in the organisation led to the study of two main questions: 1) Why did the earlier attempts not succeed? and 2) What would be a working solution?

The clarification of past experiences and of the current status of knowledge capturing and sharing means and activities was carried out by interviewing employees and analysing relevant documents. This brought forward that both managers and designers felt that a lot of knowledge was being wasted. Existing knowledge was difficult to find and when found it was not in a reusable form.

The underlying organisational goal was to reduce software defects by increasing the knowledge transfer between different projects. The information to be shared was stored in the Lessons to Learn Database. Interviews indicated clearly that project managers' awareness of the database was very low, and its use poor. An analysis of the database revealed that there were a number of incomplete entries and only one of the four thematic sections was in active use. According to the database concept owner, the reason was that the preparation of database entries required a lot of effort and that administering the data was difficult. Moreover, the accuracy and relevance of the data was not always obvious, because “there has to be a balance between enabling free expression and maintaining control”.

Data Transfer Days was another form of sharing knowledge between projects. Their purpose was twofold: to identify and to analyse problems faced in the projects in order to avoid them in the future, and to analyse past success stories. These meetings were unanimously regarded useful. Important knowledge was captured and shared, even though people found it hard to remember past successes and pitfalls once their projects had ended. The results of these events were planned to be analysed, packaged and followed-up for new projects, but, unfortunately, at this point enthusiasm usually disappeared. The meetings were useful mainly for those who were able to attend. Nevertheless, free face-to-face conversation between group members turned out to be a better way of sharing knowledge than the database (notification supported by Davenport and Prusak 1998 as well).

In summary of the current status, neither the Data Transfer Days nor, in particular, the Lessons to Learn Database was working as initially intended. The reason for the latter was that neither filing nor searching of knowledge was incorporated into the project processes. An efficient use of the database would have required more disciplined processes and a lot more effort at capturing, packaging, searching, maintaining and reusing the knowledge. Furthermore, most projects were too busy with coping with their everyday problems to be willing to contribute to any additional overhead duties. In general, software designers tended to trust anyone nearby, rather than experts or even less the shared database.

8.2.2 Need Based Experience Capturing Process

Based on the problems discovered while analysing the current status of knowledge reuse several process improvement actions could have been chosen. For example, software process management could have been improved by defining additional processes to search and update the Lessons to Learn Database or to improve analysis and packaging of the results of Data Transfer Days or to use training as a means to build awareness of existing reuse possibilities. Eventually, improvement actions that would add duties to software managers and engineers was not favoured for the following reason: knowledge searching and capturing was seen by software managers as extra work. Busy with their daily duties, they expected to be served by knowledge they needed. The favouring of knowledge pushing is also supported, e.g., via Henninger's and Schlabach's (2001) alert system. Moreover, NASA's Goddard Space Flight Center has reported building a user-profile-based push feature to shift from passive to active dissemination of lessons learnt and thus facilitate better knowledge sharing (Liebowitz 2002).

To summarise, the goal for the new solution was that it should affect the software development projects and processes as little as possible and not require new technologies. The goal was to establish a process for capturing experience by SPI experts as knowledge capturing agents. Thus the aim of this SPI action was not to create or enhance any KM system, but rather a process that would help to acquire experience from existing sources, such as the company's databases and individual persons, based on the identified needs of ongoing software engineering projects.

Since the existing processes should not be touched, simple manual off-line means were preferred. The software development projects were regarded as a customer, who are to be served by the knowledge they need as an opposite to a large-scale acquisition, analysis, packaging, sharing and updating of knowledge for projects to come. A process model for capturing knowledge by using SPI experts was established. The new model consisted of the Knowledge Capturing

Project and Customer Projects, which stated the needs for experience capturing. The Capturing Project was responsible for gathering knowledge from relevant sources, packaging and providing it to a Customer Project to use. The solution did not change the organisational setting, nor did it require new tools. Knowledge was expected to be found from existing sources, such as project final reports, error databases, discussion forums, and, most importantly, from people. The experience capturing process itself consisted of three separate base practices: (1) definition of scope and requirements for knowledge capturing, (2) knowledge acquisition, and (3) packaging knowledge. These processes were further divided into sub phases with detailed process descriptions with goal statements, inputs, entry and exit criteria and so on to assist the capturing project to create an experience package that fulfilled the knowledge needs of a customer project. A simplified capturing process is illustrated in *Figure 28*.

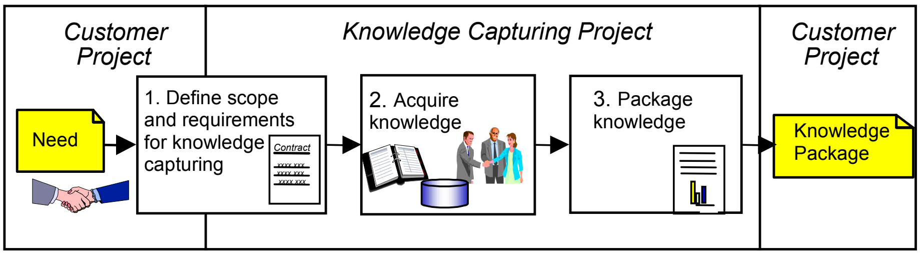

Figure 28. The simplified knowledge capturing process (Paper VII).

The developed approach was tested in an industrial environment. The needs of the customer project were structured to indicate what knowledge was required, in what form, and how it was intended to be reused. The delivered interface-related knowledge package met the requirements set for it completely. The approach proved to work well, and the project was served just in time with the knowledge package they needed.

8.2.3 Conclusions

Product development goals and the expected value of reuse of design expertise drove the KM based SPI effort. The organisation's autonomous project managers were committed to the short-term goals of producing a specific product with required features within determined schedule. They did not have time either to seek extensively for stored knowledge, or to provide it to others.

They needed to be served with relevant knowledge in the right form and at the right time. For this reason the earlier database-driven approach where the software engineers and managers had been responsible for acquiring and maintaining the knowledge did not succeed. Knowledge sharing events between group members had proven to be a better way of sharing knowledge than the database. Nevertheless, the plan to package the results of these events for reuse in forthcoming projects never materialised. To succeed, the SPI organisation had to take the leading role of knowledge acquirer and broker. The SPI-based KM became, to a large extent, a need-based service for software development projects.

8.3 Evaluation

Where the previous method development research cycles proposed the Pr^{2}imer and Profes improvement methods the fourth cycle concentrated on studying and understanding the KM problems and constraints companies may have. Based on this study, the need-based knowledge capturing process was defined and executed in an industrial environment. In this KM-based SPI, the improvement approach proposed by Pr^{2}imer and enhanced by Profes was applied successfully. The new knowledge capturing process improved software engineering by capturing and offering the required technical type of information for the project. Using knowledge that was built on this information, the project was better able to continue with technical planning. Thus, the result of this engineering research cycle is not a new SPI method as such but rather an instantiation of earlier developed SPI methods caring for a need-based knowledge capturing process serving projects on a case-by-case basis. Because of the nature of the fourth engineering research cycle, the CSF evaluation is not utilised.

8.4 Summary

The aim of this research was to clarify how software engineering knowledge acquisition and sharing could be supported by using SPI principles found critical for success.

The lessons learnt from building the experience capturing process again raised again a flag for clarifying the existing, actual needs for action and, based on that, seeking a solution. Attempts to manage software engineering knowledge, to search for and share it extensively, turned out to be a rocky and resource-consuming approach. The developed and piloted need-based knowledge capturing process is an example of just-in-time service for a project that critically needed existing technical knowledge about software and system interfaces, knowledge that directly affected the quality of project's design phase. The needed knowledge was located in various sources and thus difficult to capture. The fact that SPI experts successfully took over this knowledge management problem advocates the supportive and assisting role an SPI organisation should have regarding software development projects.

9. Summary and Conclusions

In the previous chapters three research questions posed in Chapter 1 have been studied. Chapter 1 introduced the background of this research and depicted the research setting and the author's contribution to the research. Chapter 2 provided an overview to various SPI research results that are relevant to this research. In SPI method development research, many of them are attached to the new or further enhanced SPI methods. Chapter 3 captured SPI lessons learnt from literature and developed the CSF criteria that was first used in Chapter 4 to evaluate related research. Chapter 5 presented and evaluated an integrated improvement management method Pr^{2}imer. Chapter 6 enhanced the role of measurement as an important part of an SPI method. Based on experiences and new ideas, further development and enhancement of the SPI method was recapitulated and evaluated in Chapter 7. This product quality based process improvement method is called Profes. In Chapter 8 the use of KM in SPI was discussed and one practical and tested solution for capturing knowledge was presented. In this Chapter 9 the research results are recapitulated in the light of the research questions; moreover, further research is discussed.

9.1 Answers to Research Questions

The research results presented in this thesis are based on the author's long-term involvement in several industrial SPI projects and in SPI method development according to the engineering research method (Adrion 1993, Glass 1994).

The main research question studied was *"How to develop and evaluate industrial SPI methods?"* To be better able to find an answer to this question it has been divided into three sub questions that have been studied in four method development research cycles in various industrial settings.

Q1. What are the most typical industrial SPI needs regarding the SPI methods? The means to seek answer to this questions are presented in connection with the third research question. The extracted needs are condensed into the following 15 propositions that are organized under management, commitment and cultural related subheadings, and further to four improvement engineering related steps: plan, do, check and act according to the PDCA cycle. This body of propositions form a framework any SPI method should be able to answer in order to maximise possibilities for success and minimise risks for failures.

<u>Improvement management</u>

1. An SPI method supports different SPI approaches.
2. An SPI method supports active participation of all affected parties.
3. An SPI method supports co-operation with software engineers.
4. An SPI method supports training being planned within and as a part of the initiative

<u>Commitment</u>

5. An SPI method supports commitment of top managers.
6. An SPI method supports commitment of middle managers.
7. An SPI method supports commitment of software engineers.

<u>Cultural aspect</u>

8. An SPI method supports developing improved solutions on a case-by-case basis.

<u>Plan</u>

9. An SPI method supports clarifying the current status of processes.
10. An SPI method supports establishing a link between business goals and improvement goals.
11. An SPI method supports measurable improvement goals.
12. An SPI method supports generating an improvement plan.

<u>Do</u>

13. An SPI method supports developed solutions are tested before large-scale use.

<u>Check</u>

14. An SPI method supports using metrics in monitoring improvement actions and results.

<u>Act</u>

15. An SPI method supports the sustainability of an improvement initiative.

Q2. What kinds of SPI methods are suited to these needs? The answer to this question has been researched via four method development research cycles. The analysis of various existing methods provides the understanding that none of them was able to meet the needs industry had, but that wisely combining and enhancing them should lead to better results. Chapters 5 - 8 presented results of this work. **First**, Chapter 5 presented and evaluated the SPI method, Pr^{2}imer, that builds on the ideas of the PDCA cycle (Deming 1986) and teamwork and integrates software process analysis methods, process modelling and measurements into one complex. **Second**, the research interest was in measurement activities that were enhanced and automated (Chapter 6). **Third**, changing the improvement initiator from software process quality to product quality according to business needs was studied and the research result was packaged in the form of the Profes improvement methodology (Chapter 7). Profes builds on the same improvement principles as Pr^{2}imer enhanced with measurements further packaging the approach to a cookbook type of handbook. Furthermore, Profes distinguishes the improvement effort on organisational and project levels according to the QIP principles (Basili et al. 1994b), makes the commitment process explicit, and introduces the new concept of Product-Process-Dependency. **Last**, the question of capturing knowledge that can be used to improve software development was studied in an industrial setting and the developed knowledge capturing process was described in Chapter 7.

Q3. How to gather and analyse the practical experiences of SPI methods in order to develop them further? There were several ways to seek answers to this question. **First**, industrial needs were extracted using literature survey researching references describing industrial SPI cases or reporting larger surveys of SPI experiences. The lessons learnt from single cases were then grouped and presented as a group of propositions relevant to any SPI method. The result of this study was encapsulated in Chapter 3 as 15 success factor propositions that raise the possibility of successful SPI. These propositions formed a basis for CSF evaluation that was first used to evaluate the related research (Chapter 4). Furthermore, this CSF evaluation was used to evaluate the results of method development research cycles in Chapters 5, 6, and 7. **Second,** the requirements for a product-quality-focused method development cycle were clarified by enquiring among three organisations offering their software development environment for method development and validation. Examples of requirements stated by these organisations are as follows “the methodology should consist of a number of building blocks, from which specific items could be selected; the methodology should be open (e.g. exchangeable with respect to different basic methodologies, such as ISO9126 versus TQM); the methodology should support bottom-up as well as top-down process improvement (where bottom-up is preferred)” (Paper VI). **Last,** co-operating with organisations and software project teams in practice provided an opportunities to apply the methods and thus to follow the actual improvement work and effects of improvement actions, and to gain immediate feedback. Moreover, it afforded the opportunity to accumulate knowledge and understanding of industrial needs. Most of these experiences are described in Paper II.

Table 14 presents how the research focus changed during the method development research cycles. Legend "XXX" indicates that the area was a target of research, "X" means that the area was part of the research, and "–" shows an area that was not yet studied.

Table 14. The changes in research focus.

The extension of Research focus	SPI Management	Process Quality		Measurement	Product Quality	KM
		Appraisa	Standard			
The first cycle	XXX	X	X	X	–	–
The second cycle				XXX	–	–
The third cycle	X	X		X	XXX	X
The fourth cycle						XXX

The changes in research focus show the learning process as well. First, the main goal was to find the basic structure in which SPI should be executed. Other elements understood to be important were the use of various means to clarify the current status of a process (process quality focus) and the importance of measurement to support SPI. The measurement was the focus area studied in more detail during the second engineering research cycle. At that time, no explicit attention was given to knowledge management or product quality as an improvement initiator, which became the focus in the third cycle. By that time, it was already known how SPI should be executed, but the lack of business focus became obvious. Companies do not do business with good processes but with the products they sell. This understanding understandably boosted the need to change the improvement focus from processes to a product. By this time, organisational environment and software development projects became more and more complex, raising the question of how to manage the extensive information related to SPI. This study is still going on.

9.2 Limitations of the Results

This thesis provides an extensive description of the development and maturation of SPI methods during one decade. The development path has been evolutionary by nature; each method engineering cycle has augmented the results of previous cycles. The research results are culminated in two SPI methods, Pr^{2}imer and Profes. In this thesis, also CSF criteria are developed for evaluating SPI methods.

Pr^{2}imer is the first comprehensive SPI method that unites different SPI approaches into a single ensemble. It has been developed and applied in a variform embedded software development environment. Despite its large coverage, Pr^{2}imer is capable of remaining a simple and easy to follow improvement model. Pr^{2}imer does not provide any extensive explanations of improvement steps, but rather captures improvement principles and states outputs of each improvement phase. Even though Pr^{2}imer has been developed for the improvement of embedded software, it can be easily applied to any SW development as it does not contain any specific embedded software development features. The limitation of Pr^{2}imer is that it provides no guidance for organisational improvement infrastructure, which narrows its application scope mainly to the software project level.

The Profes methodology is presented carefully and in detail with a wide selection of resources, tools, methods, and templates available for supporting the improvement work. Profes covers and unites elements regarded as important for SPI, such as process assessment, product assessment, product quality-process dependency, process modelling, measurement, and learning form experience. Profes also discusses how to build the infrastructure for SPI and also presents some advanced improvement techniques to be applied. This broad scope combined with the detailed presentation format of Profes, however, also shows some drawbacks; it may easily be classified as a heavy approach to SPI. Yet, it has to be noted that Profes, as well as Pr^{2}imer, has been developed in a software development environment in which legacy embedded software is developed and maintained. Although this gives the primary application area for both methods, it does not delimit the scope to legacy systems only. Pr^{2}imer can be seen as a light weight version of Profes, which was also developed to be tailorable and adjustable. For example, the Profes Step "Gain commitment" guides and instructs commitment retrieval and can thus also be employed as an augmentation to any SPI method. This applies to many other steps of Profes as well.

In this thesis, the CSF criteria set is developed and used for evaluating SPI methods. CSFs are based on a synthesis of a literature review covering mainly SPI case results focusing on factors that are important for successful SPI. Thus, it is expected that the more CSFs are fulfilled, the lower the risk of failure and the higher the possibility of success. Although CSFs contain propositions that are in general considered to have a positive effect on improvement success, it is not claimed that fulfilling all these will inevitably lead to success. This fact also reveals a limitation of the CSF criteria; the criteria have to be applied by interpreting the improvement context. Ultimately, to be able to make a well reasoned and conscious decision over relevant and irrelevant factors, it is crucial to understand the general SPI success factors. The improvement context will dictate which CSFs are to be highlighted and which are less important, or even if they should be considered at all.

To summarize, despite all methodological and tool support, SPI remains, significantly, an intellectual activity. Even the best fitted SPI methods do not sway this fact, but still they provide essential and needed support and guidance for SPI.

9.3 Further Research

This thesis has presented a cross-section of SPI improvement method development and evaluation conducted during one decade. Still, there remains room for further research, development and validation. More research is needed to explore how to take improvement results in use. While the first phases of SPI initiative are generally known, and there are several ways of determining “what to improve”, the question “how to improve” has been researched far less, thus requiring more attention in the future.

The concern with the continuous learning aspect of SPI has only begun to mature as a research area within SPI. KM support for continuously improving the execution of SPI and software engineering is an area not yet extensively explored or structured. Steps in this direction have been taken (cf. Dingsøyr 2002, Pourkomeylian 2002, Lindvall & Rus 2003), but a lot of research is still needed to create practical and tested solutions for KM supported SPI.

This research introduces SPI methods developed in environments where the software development has been traditional comprising software projects that last at least half to one year and are often related to the development of legacy systems as well. Thus, the research results presented in this thesis fit, unavoidably, best to the software development of this type. The growing need to react quickly to the changing customer requirements has boosted the era of new software process development methods referred to as Agile methods (Abrahamsson et al. 2002). The one further research areas arise from these changes in software development business strategies that are supported with new software development methods. In Agile software development environments, the role of quality software development process in relation to quality of product is understood and handled differently from, for example, the development of legacy systems. It may be asked, what the role of SPI within Agile software development would be, and furthermore, whether improvement needs of this type of software development can be directly addressed with any of the existing SPI methods.

10. Introduction to the Original Papers

This Chapter gives an overview of the original papers constituting the basis of this dissertation. In the following sections, the content of the papers is discussed. *Table 15* presents basic information of the papers and shows how each one of them contributes to the research questions studied (Q1, Q2, and Q3).

Table 15. The original papers of the thesis in relation to the research questions.

Related Original Papers	Published in	Forum	Q1	Q2	Q3
Paper I	1996	Quality Engineering, Vol. 8		✓	
Paper II	1998	EuroSPI	✓		✓
Paper III	1998	SPI'98		✓	
Paper IV	2001	Metrics Symposium		✓	
Paper V	1998	SPI-98		✓	
Paper VI	2000	Profes	✓		✓
Paper VII	2002	IEEE Software May/June 2002		✓	✓

10.1 Towards an Integrated SPI method

10.1.1 Paper I

Paper I introduces the main ideas, principles, and basic functionality of Pr^{2}imer, which as the first practical SPI method unites various SPI approaches – process analysis, measurement and process modelling – to a functional ensemble. The paper also presents the results of a case study conducted at the leading manufacturer of medical instruments. The main considerations of the paper are the following:

- Quantitative and/or qualitative analysis of current software development practices,
- Definition of measurable goals for improvement,
- Planning of successive and practical process improvement steps, and
- Piloting and trial use of the new practices.

10.1.2 Paper II

Paper II recapitulates the current status of Pr^{2}imer method development, the main part of the paper summarising the experiences gained and SPI lessons learnt from applying Pr^{2}imer method to over 20 SPI cases in the course of five years. Among the key observations presented in the paper are:

– Top management commitment guarantees the resources required for improvement work at software engineering level,

– Current state analysis using qualitative methods has solely provided a good basis for improvement,

– Current state analysis does not reveal any unknown problems, but it documents the situation and provides a basis for improvement discussions,

– People holding various positions in the organisation should be involved in improvement goal definition and improvement planning,

– To enable follow-up and evaluation, SPI should always be accompanied with appropriate measurements, and

– Support and feedback to the projects have to be given regularly. The GQM feedback session provides a good forum for this.

10.2 Enhanced Role of Measurement

10.2.1 Paper III

Paper III reinforces Pr^{2}imer with enhanced goal driven measurement activities and applies the improved method to testing and requirements engineering processes. The paper proposes a way to unite the GQM and improvement processes as described in Pr^{2}imer. It was noted that the act of taking new measurement practices in use alone would improve the used practices. The first time of testing measurement automation using the MetriFlame tool yielded promising results, which led to studying the possibilities to automate measurement data collection and analysis. The two case studies discussed in the paper originate from ABB Transmit Oy Relays and Network Control, and Valmet Automation.

10.2.2 Paper IV

Paper IV proposes a measurement automation process based on goal-driven measurement. Furthermore, the measurement automation process is presented and linked to Pr^{2}imer and GQM processes. The paper sets general requirements for automated measurement tool support, describes the metrics automation process in detail, discusses experiences of automation, and puts forward the following points:

- Measurement automation provides many advantages, while simultaneously requiring planning before automation can take place,
- Data analysis, data sources and scheduling have to be planned in detail, so as to allow the selected metrics to be automated,
- The optimum for measurement automation would be that an organisation already had measurement practices in place, and
- Technical solutions and tools play a remarkable role in measurement automation.

10.3 Product Quality Focused Improvement

10.3.1 Paper V

Paper V describes a change in the improvement strategy. The paper presents how to move the improvement basis beyond processes, to the quality of the software. It describes Profes, the product quality driven improvement methodology, which supplements the Pr^{2}imer method with a product focus. The main sections of the paper are concerned with:

- Introduction to the background elements
- Description of the Profes improvement methodology
- An example of building and using PPD models in product quality driven SPI.

10.3.2 Paper VI

Paper VI recapitulates the status and design rationale of the Profes improvement methodology. The enabling technologies for SPI include the following: QIP, software process and product assessment, Goal-Oriented measurement, process modelling, and know-how reuse.

10.4 Managing SPI Knowledge

10.4.1 Paper VII

Paper VII recalls the old lesson learnt and highlights the necessity of a need-based approach in the area of KM driven SPI. The paper discusses unsuccessful attempts of using a general "Lessons Learnt" database in capturing and sharing software engineering related knowledge. Based on the analysis of these attempts, a need-based knowledge capturing process is defined and piloted to gather information required for a specific software development project. The paper also recapitulates the background of the KM supported SPI work and describes the developed solution.

References

Abrahamsson, P. 2002. The Role of Commitment in Software process Improvement. Acta Univ. Oul. A 386. 158 p.

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. 2002. Agile Software Development Methods, Review and Analysis. VTT Publications 478. Espoo, Finland. 107 p.

Adrion, W. R. 1993. Research Methodology in Software Engineering. Workshop on Future Directions in Software Engineering, Software Engineering Notes, Summary of the Dagstuhl. Vol. 18, No. 1.

Bang, S. 2000. CMM Improvements in the fast lane. In the Workshop of EuroSPI‘2000 - European Software Process Improvement. Copenhagen Business School. November 7–9, 2000. Copenhagen, Denmark.

Barnard, J. & Price, A. 1994. Managing Code Inspection Information. IEEE Software. March 1994. Pp. 56–69.

Basili, V. R. 1993. Applying the Goal/Question/Metric paradigm in the Experience Factory. In the 10th Annual CSR Workshop, October 1993. 23 p.

Basili, V. R., Caldiera G. & Rombach, H. D. 1994a. Goal Question Metric Paradigm. Encyclopaedia of Software Engineering. Vol. 1 ed. John Wiley & Sons. Pp. 528–532.

Basili, V. R., Caldiera G. & Rombach, H. D. 1994b. Experience Factory. Encyclopaedia of Software Engineering. Vol. 1. ed. John Wiley & Sons. Pp. 469–476.

Basili, V. R. & Caldiera, G. 1995. Improve Software Quality by Reusing Knowledge and Experience. Sloan Management Review/Fall. Pp. 55–64.

Basili, V. R. & 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. & Seaman, C. 2002. The Experience Factory Organization. IEEE Software May/June 2002. Pp. 30–31.

Basili, V., Tesoriero, R., Costa, P., Lindvall, M., Rus, I., Shull, F. & Zelkowitz, M. 2001. Building an Experience Base for Software Engineering: A Report on the First CeBASE eWorkshop. In the proceedings of Third International Conference of Product Focussed Software Process Improvement. Profes 2001. Eds. Bomarius, F. & Komi-Sirviö, S. Kaiserslautern, Germany, September 10–13, 2001. Pp. 110–125.

Basili, V. R. & Turner, A. 1975. Iterative Enhancement: A Practical Technique for Software Engineering. IEEE Transactions on Software Engineering, Vol. 1, No. 4, 1975. Pp. 390–396.

Basili, V. & Weiss, D. M. 1984. A Methodology for collecting valid software engineering data. IEEE Transaction on Software Engineering, Vol. 10, No. 6. Pp. 728–738.

Baskerville, R. 1996. Structural Artifacts in Method Engineering: The Security Imperative. In the Proceedings of the IFIP TC8 Working Conference on Method Engineering: Principles of method construction and tool support (eds. Brinkkemper, S., Lyytinen, K., & Welke, R.) Chapman & Hall, Great Britain. Pp. 8–28.

Bazzana, G. & Fagnoni, E. 1999. Process Improvement in the Internet Service Providing. Eds. Messnarz, R. & Tully, C. Better Software Practice for Business Benefit. Principles and Experiences. IEEE Computer Society. Pp. 267–279.

Beck, K. 1999. Extreme Programming Explained: Embrace Change. Addison-Wesley. ISBN 0-201-61641-6. 224 p.

Bicego, A., Derks, P., Kuvaja, P. & Pfahl, D. 1999. Product Focussed Process Improvement: Experiences of Applying the Profes Improvement Methodology at Dräger. In the Proceedings of Euromicro'99 Conference. September 1999, Milan Italy. Pp. 55–68.

Bicego, A., Khurana, M., Kuvaja, P. & Lehtonen, J. 1998. SPICE conformant assessment method for embedded systems, the Profes project report 4.2.B-I. Version 1.1. September 9, 1998. 26 p.

Birk, A. & Tauz, C. 1998. Knowledge Management of Software Engineering Lessons Learned. In the proceedings of the Tenth Conference on Software Engineering and Knowledge Engineering (SEKE 1998). United States of America, Illinois, Skokie. Knowledge Systems Institute. Pp. 24–31.

Birk, A., Derks, P., Hamann, D., Hirvensalo, J., Oivo, M., Rodenbach, E., van Solingen, R. & Taramaa, J. 1998. Applications of Measurement in Product-Focussed Process Improvement: A Comparative Industrial Case Study. In the Proceedings of the 5th International Metrics Symposium (Metrics'98). Bethesda, MD. November 20–21, 1998. Pp. 105–108.

Birk, A., Dingsøyr, T. & Stålhane, T. 2002. Postmortem: Never Leave a Project without It. IEEE Software May/June 2002. Pp. 43–45.

BNQP 2003. Malcolm Baldrige National Quality Award. http://www.quality.nist.gov/About_BNQP.htm. Accessed: August, 20, 2003.

Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., MacLeod, G. & Merrit, M. J. 1978. Characteristics of Software Quality. Vol. 1. TRW Series of Software Technology. Amsterdam, Netherlands.

Boehm, B. W. 1988. A Spiral Model for Software Development and Enhancement. IEEE Computer, Vol. 21, No. 5. Pp. 61–72.

BootCheck 1997. BootCheck-käyttöohje Versio 1.0. 1997. Nuotio-projekti. 14 p. (in Finnish).

Briand, L., Differding, C. & Rombach, D. 1997. Practical Guidelines for Measurement-Based Process Improvement. Software Process Improvement and Practice Journal, Vol. 2, No. 4. 1997, pp. 253–280.

Bryant, A. 2002. Grounding Systems Research: Re-establishing Grounded Theory. In the proceedings of the 35th Hawaii International Conference on Systems Sciences. January 7–10, 2002. Big Island, Hawaii. Pp. 253c–263.

Burgess, A. 1995. Mad about or Mad at Measurement. IEEE Software, January 1995. Pp. 115–116.

Card, D. 1991. Understanding Process Improvement. IEEE Software. July 1991. Pp. 102–103.

Card, D. 1993. What Makes for Effective Measurement. IEEE Software. November 1993. Pp. 94–95.

Cockburn, A. 2001. Agile Software Development. Addison-Wesley. Boston. ISBN 0201699699. 256 p.

CMMI. 1999. Capability Maturity Model - Integrated Systems/Software Engineering, Continuos representation. CMU/SEI. Public-Released Draft. Vol. 1, Version 0.2b.

CMMI. 2000. CMMI^{SM} for Systems Engineering/Software Engineering, Version 1.02. CMU/SEI-2000-TR-028. Carnegie Mellon, Software Engineering Institute.

Conradi, R. & Fuggetta, A. 2002. Improving Software Process Improvement. IEEE Software. July/August 2002. Pp. 92–99.

Curtis, B., Krasner, H., Shen, V. & Iscoe, N. 1987. On Building Software Process Models under the Lamppost. Proceedings of the 9th International Conference on Software Engineering (ICSE 1987). IEEE Computer Society Press. Monterey. CA. Pp. 96–103.

Daskalantonakis, M. K. 1994. Achieving higher SEI levels. IEEE Software. July 1994. Vol. 11 Issue 4. Pp. 17–24.

Davenport, T. H. & Prusak, L. 1998. Working Knowledge: How Organizations Manage What They Know. Boston, United States of America. Harvard College Business School Press, 1998. 199 p.

Debou, C., Courtel, D., Lambert H., Fuchs, N. & Haux, M. 1999. Alcatel's Experience with Process Improvement. Eds. Messnarz, R. & Tully, C. Better Software Practice for Business Benefit. Principles and Experiences. IEEE Computer Society. Pp. 281–301.

Delmiglio, R., Di Muro, S., Humml, S., Lora, A., Bazzana, G. & Rumi, G. 1999. From Assessment to Improvement. An Experience in the GSM Application Domain. Eds. Messnarz, R. & Tully, C. Better Software Practice for Business Benefit. Principles and Experiences. IEEE Computer Society. Pp. 237–265.

Deming, W. E. 1982. Methods for Management of Productivity and Quality. George Washington University, Washington, D.C.

Deming, W. E. 1986. Out of the Crises: Quality, Productivity and Competitive Position. MIT Center for Advanced Engineering Study. Cambridge, MA.

Diaz, M. & Sligo, J. 1997. How software process improvement helped Motorola. IEEE Software, September-October 1997. Vol. 14, Issue 5. Pp. 75–81.

Dingsøyr, T. 2002. Knowledge Management in Medium-Sized Software Consulting Companies. Norwegian University of Science and Technology. Tapir Trykkeri. Trondheim. 238 p.

Drouin, J.-N. 1999. The SPICE Project. In Elements of Software Process Assessment and Improvement. Eds. El Emam, K. & Madhavji, N. H. IEEE Computer Society. Los Alamitos, California. 1999. ISBN 0-8186-8523-9. 384 p.

Dutta, S., Lee, M. & Van Wassenhove, L. 1999. Software Engineering in Europe: A Study of Best Practices. IEEE Software May/June 1999. Pp. 82–89.

Dybå, T. 2000. An Instrument for Measuring the Key Factors of Success in Software Process Improvement. Empirical Software Engineering, 5. 2001 Kluwer Academic Publishers, Boston. 2000. Pp. 357–390.

Dyne, K. 1998. Ericsson: Benefits, The Ericsson Strategic Software Initiative. In the proceedings of the 3rd Annual European Software Engineering Process Group Conference 1998. London, UK. June 8–11, 1998.

EFQM 2003. European Quality Awards. http://www.efqm.org/welcome.htm. Accessed: August 20, 2003.

EIA/IS-731. 1998. Electronic Industries Association. Systems Engineering Capability Model. Washington, DC. http://www.geia.org/eoc/G47/page6.htm. Accessed: August, 20, 2003.

El Emam, K., Drouin, J. & Melo, W. 1998. SPICE, The Theory and Practice of Software Process Improvement and Capability Determination. IEEE Computer Society Press. 486 p.

El Emam, K., Fusaro, P. & Smith, B. 1999. Success Factors and Barriers for Software Process Improvement. Eds. Messnarz, R. & Tully, C. Better Software Practice for Business Benefit. Principles and Experiences. IEEE Computer Society. Pp. 335–371.

ESA. 1994. ESA PSS-05-0 Software Engineering Standards. European Space Agency. Issue 3. 130 p.

Fenton, N. & Pfleeger, S. 1999. Software Metrics: A Rigorous & Practical Approach. Second edition. PWS Publishing Company, Boston. ISBN 0-534-95425-1.

Fitzgerald, B. & O'Kane, T. 1999. A Longitudial Study of Software Process Improvement. IEEE Software May/June 1999. Pp. 37–45.

Florac, W. A. & Carleton, A. D. 1999. Measuring the Software Process. Statistical Process Control for Software Process Improvement. Massachusetts. Addison-Wesley Longman, Inc. 250 p.

Florac, W. A., Park, R. E. & Carleton, A. D. 1997. Practical Software Measurement: Measuring for Process Management and Improvement. CMU/SEI-97-HB-003. 246 p.

Fuggetta, A. 2003. Open source software - An evaluation. Journal of Systems and Software, Vol. 66, Issue 1. April 15, 2003. Pp. 77–90.

Gilb, T. 1988. Principles of Software Engineering Management. Addison-Wesley. Reading. MA. ISBN 0-201-19246-2. 442 p.

Glass, R. L. 1994. The Software Research Crisis. IEEE Software November 1994. Pp. 42–47.

GMOD 1992. V-Model: Software Lifecycle Process Model, General Report No. 250, German Ministry of Defence.

Goldenson, D. R. & 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. Software Engineering Institute. Pittsburgh. 50 p.

Grady, R. B. 1992. Practical Software Metrics for Project Management and Process improvement. P T R Prentice Hall, Inc. Englewood Cliffs, New Jersey 07632. ISBN 0-201-60444-2. 270 p.

Gresse, C., Hoisl, B. & Würst, J. 1995. A Process Model for GQM-Based Measurement. STTI-Report. University of Kaiserslautern. 229 p.

Haley, T. 1996. Software process improvement at Raytheon. IEEE Software. November, 1996. Pp. 33–41.

Hall, T., Baddoo, N. & Wilson, D. 2001. Measurement in Software Process Improvement Programmes: An Empirical Study. In the proceedings of IWSM 2000. Springer-Verlag Berlin Heidelberg 2001. LNCS 2006, Pp. 73–82.

Halstead, M. H. 1977. Elements of Software Science. Elsevier, New York, 1997.

Halvorsen, C. P. & Conradi, R. 2001. A Taxonomy to Compare SPI Frameworks. V. Ambriola (ed.). In the proceedings of Software Process Technology, the 8th European Workshop, EWSPT 2001, LNCS 2077, Witten, Germany, June 19–21. Pp. 217–235.

Hamann, D., Järvinen, J., Birk, A. & Pfahl, D. 1998. A Product-Process Dependency Definition Method. In the proceedings of the Euromicro Workshop on Software Process and Product Improvement. Ed. Chroust, G. Västerås Sweden. August 25–27, 1998. Pp. 898–904.

Haug, M., Olsen, E. W. & Consolini, L. 2001. Software Quality Approaches: Testing, Verification and Validation. Software Best Practice 1. ESSI Practitioners' Reports. Springer-Verlag Berlin Heidelberg New York. 302 p.

Heath, H. & Cowley, S. 2004. Developing a grounded theory approach: a comparison of Glaser and Strauss. International Journal of Nursing Studies, Vol. 41. Pp. 141–150.

Heijstek, A. 1998. Ericsson Netherlands Efforts to Reach CMM Level 4. In the Proceedings of The European Conference on Software Process Improvement, SPI 1998. December 1–4, 1998. Monte Carlo. 1 p.

Henninger, S. & Schlabach, J. 2001. A Tool for Software Development Knowledge. In the proceedings of Third International Conference of Product Focussed Software Process Improvement. Profes 2001. Eds. Bomarius, F. & Komi-Sirviö, S. Kaiserslautern, Germany, September 10–13, 2001. Pp. 182–195.

Herbsleb, J., Carleton, A., Rozum, J., Siegel, J. & Zubrow, D. 1994. Benefits of CMM-based Software Process Improvement: Executive Summary of Initial Results. Special report. CMU/SEI-94-SR-013. September 1994. 16 p.

Hollenbach, C., Young, R., Pflugard, A. & Smith, D. 1997. Combining Quality and Software Improvement. Communications of the ACM, Vol. 40, No. 6. Pp. 41–45.

Humphrey, W. S. & Sweet, W. L. 1987. A Method for Assessing the Software Engineering Capability of Contractors. Preliminary Version. CMU/SEI-87-TR-23. Software Engineering Institute, Carnegie Mellon University. Pittsburgh, Pennsylvania.

Humphrey, W. S. 1989. Managing the Software Process. In: SEI Series in Software Engineering. Massachusetts: Addison-Wesley Publishing Company. ISBN 0-201-18095-2. 494 p.

IEEE Std 1061-1998. 1998. IEEE Standard for Software Quality and Metrics Methodology. IEEE-SA Standards Board, December 8, 1998. 20 p.

IPD-CMM. 1997. Integrated Product Development Capability Maturity Model, Version 0.98. Enterprise Process Improvement Collaboration and Software Engineering Institute. Carnegie Mellon University.

Ishikawa, K. 1985. What is Total Quality Control? The Japanese way. Prentice-Hall, Englewood Cliffs, N.J.

ISO/IEC 12207. 2002. Information Technology – Software life cycle processes. Geneva, Switzerland. 53 p.

ISO/IEC 14598-1. 1999. Information technology – Software product evaluation – Part 1: General overview. Geneva, Switzerland. 19 p.

ISO/IEC 15504-2. 1998. Information Technology – Software Process Assessment – Part 2: A reference model for processes and process capability. Technical Report type 2. CH-1211 Geneva, Switzerland. 44 p.

ISO/IEC 15504-5. 1998. Information Technology – Software Process Assessment – Part 5: An assessment model and indicator guidance. Technical Report type 2. 128 p.

ISO/IEC 15504-7. 1998. Information Technology – Software process assessment – Part 7: Guide for use in process improvement. Technical report type 2. 36 p.

ISO/IEC 15504-9. 1998. Information Technology – Software Process Assessment – Part 9: Vocabulary. Technical Report. 11 p.

ISO/IEC 9000-3. 1997. Guidelines for the application of ISO 9001:1994 to the development, supply, installation and maintenance of computer software. Internal Organization for Standardization. 32 p.

ISO 9000. 2000. Quality management systems – Fundamentals and Vocabulary.

ISO/IEC 9001. 2000. Quality management systems – Requirements. 23 p.

ISO/IEC 9126-1. 2001. Software engineering – Product quality – Part 1: Quality model. 25 p.

ISO/IEC 9126-2. 2003. Software engineering – Product quality – Part 2: External metrics. Technical report. 86 p.

ISO/IEC 9126-3. 2003. Software engineering – Product quality – Part 3: Internal metrics. Technical report. 62 p.

Jakobsen, A. B. 1998. Bottom-up Process Improvement Tricks. IEEE Software January-February 1998. Pp. 64–68.

Juran, J. M. 1999. Juran's Quality Handbook. Eds. Juran, J. M. & Godfrey, A. B. The 5th edition 1999. McGraw-Hill. ISBN 0-07-034003-X. 1872 p.

Järvinen, P. 1999. On Research Methods. Opinpaja Oy, Tampere, Finland. ISBN 951-97113-6-8. 129 p.

Kaplan, R. S. & Norton, D. P. 1996. The Balanced Scorecard: Translating Strategy into Action. Boston: Harvard Business School Press. ISBN 0875846513. 336 p.

Kasse, T. & McQuaid. 1998. Entry Strategies into the Process Improvement Initiative. Software Process- Improvement and Practice, Vol. 4. Pp. 73–88.

Kauppinen, M. & Kujala, S. 2001. Starting Improvement of Requirements Engineering Processes: An Experience Report. In proceedings of Third International Conference of Product Focussed Software Process Improvement. Profes 2001. Eds. Bomarius, F. & Komi-Sirviö, S. Kaiserslautern, Germany, September 10–13, 2001. Pp. 196–209.

Kautz, K. & Nielsen, P. A., 2000. Implementing Software Process Improvement: Two Cases of Technology Transfer. In the proceedings of the 33rd Annual Hawaii International Conference, HICS 2000. System Sciences. January 4–7, 2000. Pp. 2516–2525.

Kinnula, A. 1999. Software Process Engineering in a Multi-Site Environment, an architectural design of a software process engineering systems. University of Oulu. ISBN 951-42-5302-7. 119 p.

Kinnula, A. 2001. Software Process Engineering Systems: Models and Industry Cases. Department of Information Processing Science, University of Oulu. 2001. 115 p.

Kitchenham, B. 2000. Software Metrics: Measurements for Software Process Improvement. Blackwell, USA, 1996. ISBN 1-85554-820-8.

Kneuper, R. 2002. Supporting Software Processes Using Knowledge Management. Ed. Chang, S. K. Handbook of Software Engineering & Knowledge Engineering. Vol. 2. Emerging Technologies. World Scientific. Singapore. ISBN 981-02-4974-8. Pp. 579–606.

Knots-Q 2002. Knowledge-centered tools and methods for software production quality, the project plan. VTT Electronics. September, 2002. 17 p. Project www pages: http://www.vtt.fi/ele/research/soh/projects/knots-q/index.html. Accessed October 14, 2003.

Komi-Sirviö, S. 1995. ProMETRI - Ohjelmistoprosessin mittaaminen. Systeemityö, Vol. 2, No. 3. Pp. 24–28. (In Finnish).

Kucza, T., Nättinen, M. & Parviainen, P. 2001. Improving Knowledge Management in Software Reuse Process. In the proceedings of the Third International Conference of Product Focused Software Process Improvement, Profes 2001. Kaiserslautern, Germany, September 10–13. Springer. Pp. 141–152.

Kunzmann-Combelles, A. 1996. From Assessment to Improvement Actions. Compared Examples with CMM and SPICE Models. In the proceedings of the Fifth European Conference on Software Quality. Ireland. 1996. Pp. 60–67.

Kuvaja, P. & Bicego, A. 1993. BOOTSTRAP: Europe's assessment method. IEEE Software, Vol. 10, No. 3. May 1993. Pp. 93–95.

Kuvaja, P., Similä, J., Krzanik, L., Bicego, A., Saukkonen, S. & Koch, G. 1994. Software Process Assessment & Improvement – The BOOTSTRAP Approach. Blackwell Publishers. ISBN 0-631-19663-3. 149 p.

Känsälä, K. 1995. ProHAKE - kohti ohjelmistotuotannon parempaa hallintaa. Systeemityö, Vol. 2, No. 3. (In Finnish).

Lanzerstorfer, S. & Scherzer, H. 1999. Applying Quantitative ISO Auditing techniques – The BICO Approach. Eds. Messnarz, R. & Tully, C. Better Software Practice for Business Benefit. Principles and Experiences. IEEE Computer Society. Pp. 373–387.

van Latum, F., van Solingen, R., Oivo, M., Hoisl, B., Rombach, D. & Ruhe, G. 1998. Adopting GQM-based measurement in an industrial environment. IEEE Software January/February 1998. Pp. 78–86.

Lehman, M. M. 1995. Process improvement - the way forward. IEE Colloquium on Are Software Development Technologies Delivering Their Promise? Savoy place, London, March 21, 1995. Pp. 10/1–10/4.

Lepasaar, M., Varkoi, T. & Jaakkola, H. 2001. Models and Success Factors of Process Change. In proceedings of Third International Conference of Product Focussed Software Process Improvement. Profes 2001. Eds. Bomarius, F. & Komi-Sirviö, S. Kaiserslautern, Germany, September 10–13, 2001. Pp. 68–77.

Liebowitz, J. 2002. A Look at NASA Goddard Space Flight Center's Knowledge Management Initiatives. IEEE Software May/June 2002. Pp. 40–42.

Linders, B. 2001. Ericsson: From Staged CMM to Continuous CMMI (and back). In SPIder conference, September 25, 2001. Utrech, Netherlands. http://www.st-spider.nl/Plenary/20010925. Accessed August 21, 2003.

Lindvall, M. & Rus, I. 2003. Knowledge Management for Software Organizations. In: Managing Software Engineering Knowledge. Aurum, A., Jeffery, R., Wohlin, C. & Handzic, M. (eds.). Springer, 2003. Pp. 73–94.

Lindvall, M., Rus, I., Jammalamadaka, R. & Thakker, R. 2001. Software Tools for Knowledge Management, DACS State-of-the-Art-Report. The Data & Analysis Center for Software (DACS) is a Department of Defense (DoD) Information Analysis Center (IAC). 2001. 55 p.

Mashiko, Y. & Basili, V. R. 1997. Using the GQM paradigm to Investigate Influential Factors for Software Process Improvement. J. Systems Software 1997, Vol. 36. Pp. 17–32.

McCabe, T. J. 1976. A Complexity Measure. IEEE Transaction on Software Engineering, Vol. 2, No. 4. Pp. 308–320.

McCall, J. A., Richards, P. K. & Walters, G. F. 1977. Factors in Software Quality, Vol. 1 (AD/A-049-014 168 p.), Vol. 2 (AD/A-049-015 155 p.), and Vol. 3 (AD/A-049-055 41p). NTIS. US Rome Air Development Center Reports. Sprinfield, Virginia, National Technical Information Service.

McConnell, S. 2002. Real Quality for Real Engineers. IEEE Software, Vol. 19, Issue 2, March/April 2002 Pp. 5–7.

McFeeley, B. 1996. IDEAL^{SM}: A User's Guide for Software Process Improvement. Pittsburgh, Pennsylvania 15213: CMU/SEI-96-HB-001. 222 p.

McGuinness, É. 1996. Achieving Increases in Software Process Maturity, some Irish Case Studies. In the Proceedings of the Fifth International Conference on Software Quality. Dublin. Ireland September 16–20, 1996. Pp. 295–304.

McGuinness, É. 1999. Aiming for Increases in Software Process Maturity. Eds. Messnarz, R. & Tully, C. Better Software Practice for Business Benefit. Principles and Experiences. IEEE Computer Society. Pp. 331–353.

McGuire, E. G. 1996. Profiling Quality Management Practices in Software Process Improvement. In the Proceedings of the Fifth European Conference on Software Quality. Dublin. Pp. 50–59.

Mellor, P. 1992. Failures, faults and changes in dependability measurement. Information and Software Technology, Vol. 34, Issue 10.23, October 1992. Pp. 640–654.

Messnarz, R. & Tully, C. 1999. Better Software Practice for Business Benefit. Principles and Experiences. IEEE Computer Society. 393 p.

Mills, H. D., O'Neill, D., Linger, R. C., Dyer, M. & Quinnan, R. E. 1980. The Management of Software Engineering. IBM System Journal, Vol. 24, No. 2. Pp. 414–477.

Mowshowitz, A. 1997. Virtual Organization. Communications of the Association for Computing (ACM), Vol. 40, No. 9. Pp. 30–37.

Möller, K.-H. & Paulish, D. J. 1993. Software metrics: a practitioner's guide to improved product development. Chapman & Hall Computing. IEEE, London, Piscataway, NJ. ISBN 0-412-45900-0.

NASSCOM. 2000. Annual NASSCOM Report. Available at http://www.nasscom.org/. Accessed October 1, 2003.

Niazi, M., Wilson, D., Zowghi, D. & Wong, B. 2004. A Model for Implementation of Software Process Improvement: An Empirical Study. In proceedings of the 5th International Conference on Product Focused Software Process Improvement, PROFES 2004. Eds. Bomarius, F. & Iida, H. Kansai Science City, Japan. April 5–8, 2004. Pp. 1–16.

Niemelä, E., Kuikka S., Vilkuna, K., Lampola M., Ahonen, J., Forssel, M., Korhonen, R., Seppänen, V. & Ventä, O. 2000. Teolliset komponenttiohjelmistot, kehittämistarpeet ja toimenpide-ehdotukset. TEKES Teknologiakatsaus 89/00. Helsinki: Paino-Center Oy. ISBN 952-9621-86-8. 129 p. (Mainly in Finnish).

Niiniluoto, I. 1993. The Aim and Structure of Applied Research. Erkenntnis 38. Netherlands: Kulver Academic Publishers. Pp. 1–21.

Noguchi, J. 1995. The Legacy of W. Edwards Deming. Quality Progress, December 1995. Pp. 35–37.

Nonaka, I. & Takeuchi, H. 1995. The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation. United States of America, New York. Oxford University Press. ISBN 0-18-509269-4. 284 p.

Offutt, J. 2002. Quality Attributes of Web Software Applications. IEEE Software, Vol. 19, Issue 2, March/April 2002. Pp. 25–32.

Oivo, M. 1994. Quantitative management of software production using object-oriented models. VTT Publications 169. Espoo 1994. 72 p.

Oivo, M. & Basili, V. 1992. Representing Software Engineering Models: The TAME Goal Oriented Approach. IEEE Transactions on Software Engineering, Vol. 18, No. 10, October 1992. Pp. 886–898.

Oivo, M., Birk, A., Komi-Sirviö, S., Kuvaja, P. & van Solingen, R. 1999. Establishing product-Process dependencies in SPI. In the proceedings of European Software Engineering Process Group. European SEPG'99. Amsterdam, The Netherlands. June 7–10, 1999. 12 p.

Olve, N.-G., Roy, J. & Wetter, M. 1999. Performance Drivers. A Practical Guide to Using the Balanced Scorecard. John Wiley & Sons Ltd. Chichester. UK. 347 p.

Ould, M. A. 1996. CMM and ISO 9001. Software Process – Improvement and Practice, Vol. 2. Pp. 281–289.

Parviainen, P., Järvinen, J. & Sandelin, T. 1997. Practical Experiences of Tool Support in a GQM-based measurement Programme. Software Quality Journal, Vol. 6. Pp. 283–294.

Paulk, M. C. 1995. How ISO 9001 compares with the CMM. IEEE Software, January 1995, Vol. 12, No. 1. Pp. 74–83.

Paulk, M. C., Weber, C. V., Curtis, B. & Chrissis, M. B. 1994. The Capability Maturity Model for Software, Guidelines for Improving the Software Process. Addison-Wesley Publishing Company, Inc. 441 p.

Paulk, M., Weber, C., Garcia, S., Chrissis, M. & Bush, M. 1993. Capability Maturity Model for Software, version 1.1. SEI-93-TR-024. Software Engineering Institute.

Pfleeger, S. L. & Rombach, H. D. 1994. Measurement Based Process Improvement. IEEE Software, July 1994. Pp. 9–11.

Pfleeger, S. L., Jeffery, R., Curtis, B. & Kitchenham, B. 1997. Status Report on Software Measurements. IEEE Software March/April 1997. Pp. 33–42.

Pourkomeylian, P. 2002. Software Practice Improvement. Doctoral Dissertation. Göteborg University. ISSN 1400-741X. 142 p.

Profes PPD Repository. 1999. Understanding Patterns of Product/Process Dependence (PPD). www.iese.fhg.de/projects/profes/ppdrepository/ppdrepository.html. Accessed 05.06.2003.

Profes User Manual. 1999. Final version. 338 p. www.vtt.fi/ele/profes/index.html. Accessed 29.05.2003.

PSM – Part 1. 2000. Practical Software and Systems Measurement. A Foundation for objective Project Management. Measurement Process. Version 4.0b. Department of Defence and US Army. 28 p.

PSM – Part 4. 2000. Practical Software and Systems Measurement. A Foundation for objective Project Management. Apply Measures. Version 4.0b. Department of Defence and US Army. 52 p.

PSM – Part 7. 2000. Practical Software and Systems Measurement. A Foundation for objective Project Management. Evaluate Measurement. Version 4.0b. Department of Defence and US Army. 20 p.

Quinn, B. 1996. Lessons Learned from the Implementation of a Quality Management System to meet the Requirements of ISO 9000/TickIT in two Small Software Houses. In the proceedings of the Fifth European Conference on Software Quality. Dublin, Ireland. September 16–20, 1996. Pp. 305–314.

Rahikkala, T. 2000. Towards virtual software configuration management. VTT Publications 409. Espoo: VTT. ISBN 951-38-5567-8. 110 p. + app. 57 p.

Rapoport R. N. 1970. Three Dilemmas in Action Research. Human Relations, Vol. 23, No. 6. Pp. 499–513.

Reel, J. S. 1999. Critical Success factors in Software Projects. IEEE Software May/June 1999. Pp. 18–23.

Rifkin, S. 2001. What makes Measuring Software So Hard? IEEE Software May/June 2001. Pp. 41–45.

Rodenbach, E., van Latum, F. & van Solingen, R. 2000. SPI - A Guarantee for Success? – A Reality Story from Industry. In proceedings of Second International Conference of Product Focussed Software Process Improvement. Profes 2000. Eds. Bomarius, F. & Oivo, M. Oulu. Finland. June 2000. Pp. 216–231.

Roihu 1996. Projektisuunnitelma 1996–1997. Aliprosessien kehittäminen ja tulosten mittaaminen (Improvement and measurement of sub processes). A project plan. 10 p. 15 p. (In Finnish).

Ronkainen, J. 2003. Automatic Measurement of Change Control in a Virtual Software Corporation. Diploma thesis, Oulu University. 63 p.

Royce, W. 1970. Managing the Development of Large Software Systems: Concepts and Techniques, Proceedings of IEEE WESCON. Pp. 1–9.

Ruhe, G. 2001. Learning Software Organizations. Ed. Chang, S. K. Handbook of Software Engineering & Knowledge Engineering. Vol. 1. Fundamentals. World Scientific. Singapore. ISBN 981-02-4973-X. Pp. 663–667.

Rus, I. & Lindvall, M. 2002. Knowledge Management in Software Engineering. IEEE Software May/June 2002. Pp. 26–38.

Sanders, P. 1997. Memorandum for software management review council (SMRC). Practical Software Measurement: A Guide to Objective Program Insight. Washington. 1 p. Accessed August 29, 2003. http://www.stsc.hill.af.mil/crosstalk/1997/09/memorandum.asp.

SCAMPI 2001. Standard CMMI® Appraisal Method for Process Improvement (SCAMPI^{SM}), Version 1.1: Method Definition Document. CMU/SEI-2001-HB-001. 245 p.

Schneider, K. & von Hunnius, J.-P. 2003. Effective Experience Repositories for Software Engineering. In the Proceedings of the 25th International Conference on Software Engineering. ICSE 2003. Portland, Oregon, 3–10 May 2003. Pp. 534–539.

Schwaber, K. & Beedle, M. 2002. Agile Software Development with SCRUM. Prentice-Hall. Upper Saddle River, NJ. ISBN 6130676349. 150 p.

SCOPE. 1993. Software Certification on Program in Europe. Esprit 2 project. Http:/www.cordis.lu/. Accessed August 29, 2003.

SE-CMM. 1995. A Systems Engineering Capability Maturity Model SM, Version 1.1. CMU-SEI-95-MM-003. Software Engineering Institute.

Seppänen, V., Kähkönen, A., Oivo, M., Perunka, H., Isomursu, P. & Pulli. P. 1996. Strategic Needs and Future Trends of Embedded Software. Technology review 48/96. Tekes report.

Shewhart, W. A. 1931. Economic control of quality of manufactured product. New York: Van Nostrand.

Simon, J. 1999. Software Process Identification: A Case study Using the ISO/IEC 12207 Software Life Cycle Process Standard. Eds. Messnarz, R. & Tully, C. Better Software Practice for Business Benefit. Principles and Experiences. IEEE Computer Society. Pp. 303–315.

Sintonen, M. 1990. Basic and applied sciences – can the distinction (still) be drawn? Science Studies 3:2. Pp. 23–31.

Soihtu 1996. Projektisuunnitelma 1996–1997. Sulautettujen Ohjelmistoprosessien kehittäminen tukijärjestelmien avulla. (Improvement of embedded software development utilising support systems). A project plan. 10 p. (In Finnish).

van Solingen, R. & Berghout, E. 1999. The Goal/Question/Metric Method: A Practical Guide for Quality Improvement of Software Development. London, UK. The McGraw-Hill Companies. 199 p.

van Solingen, R. & Berghout, E. 2001. On Software Engineering and Learning Theory Facilitating Learning in Software Quality Improvement programs. Ed. Chang, S. K. Handbook of Software Engineering & Knowledge Engineering. Vol. 1. Fundamentals. World Scientific. Singapore. ISBN 981-02-4973-X. Pp. 679–696.

van Solingen, R., Derks, P. & Hirvensalo, J. 1999a. Product Focused SPI in the Embedded Systems Industry. In the proceedings of the International Conference on Product Focussed Software Process Improvement, Profes'99. VTT Symposium 195. VTT Technical Research Centre of Finland, Espoo, Finland 1999. Pp. 86–98.

van Solingen, R., Derks, P. & Hirvensalo, J. 1999b. Product Focussed Process Improvement in the Embedded Systems Industry. In the proceedings of the 10th European Software Control and Metric Conference (ESCOM'99). Herstmonceux, England. April 27–29, 1999. Pp. 345–353.

van Solingen, R., van Uijtregt, A., Kusters, R. & Trienekens, J. 1999c. Tailoring Product Focused SPI – Application and Customisation of Profes in Tokheim. In the proceedings of International Conference on Product Focussed Software Process Improvement, Profes'99. VTT Symposium 195. VTT Technical Research Centre of Finland, Espoo, Finland 1999. Pp. 363–377.

SPACE-UFO. 1998. Software Product Advanced Certification and Evaluation. Esprit 4 project number 22292. Http:/www.cordis.lu/. Accessed August 29, 2003.

Straub, D. W. 1989. Validating Instruments in MIS Research. MIS Quarterly, June 1989. Pp. 147–169.

Sulka-P 1994. Metriikkatyökalut ohjelmistotuotannossa, the Sulka-P project report, version 1.0. VTT Electronics. 44p. (In Finnish).

SWEBOK. 2001. Guide to the Software Engineering Body of Knowledge. Stone Man Trial Version 1.0 May 2001. A Project of the Software Engineering Coordinating Committee. IEEE Computer Society. 219 p.

Tanner, H. 2000. Pr^{2}imer-prosessinparantamismenetelmän kehittäminen yritys-kokemusten perusteella. Master's thesis. University of Oulu. 75 p. (In Finnish).

Tauriainen, A. 1999. Experience Capturing process and Its Enactment. Master Thesis. University of Oulu, Department of Information Processing Science. 103 p.

TickIT 2001. The TickIT guide, Executive Overview. January 2001, Issue 5.0. 6 p. http://www.tickit.org/overview.pdf. Accessed October 14, 2003.

Totem 2001. Knowledge Management Process and Methodologies and their application in Software Process Improvement, the project plan. VTT Electronics. January 2001. 27 p.

Tolvanen, J.-P. 1998. Incremental Method Engineering with Modelling Tools, Teoretical Principles and Empirical Evidence. University of Jyväskylä. Jyväskylä studies in computer science, economics and statistics 47. 301 p.

VASIE 2003. VASIE database. Maintained by Software Engineering Institute.

Wang, Y. & King, G. 2000. Software Engineering Processes, Principles and Applications. CRC Press LLC. ISBN 0-8493-2366-5. 708 p.

Watzman, A. & Perdue, J. 2002. Carnegie Mellon Announces Update of Capability Maturity Model for Software. Pittsburgh. Press Release. October 28, 2002.

Webster, J. & Watson, R. T. 2002. Analysing the Past to Prepare for the Future: Writing a Literature Review. MIS Quarterly, Vol. 22, No. 2. Pp. xiii–xxiii/June 2002.

Wei, C.-P., Hu, P. J.-H. & Chen, H.-H. 2002. Design and Evolution of a Knowledge Management System. IEEE Software May/June 2002. Pp. 56–59.

Appendix A: Success Factor Criteria

Success Factor Criteria for SPI Methods
1. Does the method support different SPI approaches?
2. Does the method support participation of all affected parties?
3. Does the method support co-operation with software engineers?
4. Does the method support planning and carrying out training as a part of the initiative?
5. Does the method support commitment of top managers?
6. Does the method support commitment of middle managers?
7. Does the method support commitment of engineers?
8. Does the method support developing improved solutions on a case-by-case basis?
9. Does the method support clarifying the current status of processes?
10. Does the method support establishing a link between business goals and improvement goals?
11. Does the method support measurable improvement goals?
12. Does the method support generating an improvement plan?
13. Does the method support developed solutions are tested in a pilot project?
14. Does the method support using metrics in monitoring improvement actions and results?
15. Does the method support the sustainability of an improvement initiative?

Appendix B: Original Success Factor Statements

Grouped Success Factors	
Commitment	References
Managers	17
Start at the top. Senior management leadership is required to launch the change effort and to provide continuing resources and priority.	Humphrey 1989
The commitment of senior management has to be 100%, and the support has to be visible	Delmiglio et al. 1999
The role of a sponsor is critical for success. SPI needs strong, visible and active senior management sponsorship with an accompanying vision, otherwise SPI gives moderate results, flags, or even a total failure.	McGuinness 1996, 1999
The leadership role, from senior management to middle management	Kunzmann-Combelles 1996
Commitment is crucial and should be present at all levels including *high-level management, project management and software engineers*.	Rodenbach et al.2000
Management commitment and a strong *project* management are key success factors.	Lanzerstorfer et Scherzer 1999
Commitment to improvement. In a company, re-organisations and changes in management were to set back the improvement program many times.	Quinn 1996
The commitment of the company to SPI activities	Lepasaar et al. 2001
Management commitment is often sought at the beginning of the improvement actions, but it should be addressed continually.	Rodenbach et al. 2000
The audits performed by a customer boost the improvement actions and help to sustain management commitment	Delmiglio et al. 1999
Ensuring management support	Kauppinen & Kujala 2001
To motivate people for continuous process improvement, a rewarding system for reported problems and suggested improvements is important.	Conradi & Fuggetta 2002
Staff and time resources should be made available to SPI	El Emam et al. 1999
Software process improvement requires investment. It takes planning, dedicated people, management time, and capital investment	Humphrey 1989
Readiness to invest resources in SPI	Lepasaar et al. 2001
Availability of company's own resources	Lepasaar et al. 2001
Distribution of information about what is going on greatly facilitates buy-in.	Lanzerstorfer & Scherzer 1999
Engineers	4
Preparing the field for improvement by personally talking to people and clarifying their aims and wishes regarding improvement facilitates future planning and makes people more committed to the forthcoming work.	Jakobsen 1998
Reviews are a very good way of spreading out knowledge and building team spirit.	Jakobsen 1998

The commitment of software engineers is important as well, and often underestimated too. There is a big risk involved in imposing fully-fledged solutions from standards and best practices to engineers.	Rodenbach et al. 2000
The commitment of engineers is achieved by introducing improvements that are based on their own ideas.	Rodenbach et al. 2000
Cultural Issues	7
The roles of cultural, learning, and long-term dimensions of SPI work are emphasised, and establishing participative engagement in all process changes is recommended.	Conradi & Fuggetta 2002
Increased emphasis on team-based, quality-focused, and process-dependent organisational culture and structure.	McGuire 1996
Organisational culture and related change management strategy coupled with appropriate training and information sessions can have a substantial effect on the rate of improvement progress	McGuire 1996
Cultural specialities at national and company levels need to be understood in order to be able to speak the same language	Debou et al. 1999
Successful SPI programs cannot be transferred as such from another culture. Different countries, companies, types of site, they all form different cultural entities.	Rodenbach et al. 2000
Change may call for a culture change in addition to changes at process or technology level. Basically, cultural changes require that the personnel understand the reason for the change.	Kauppinen & Kujala 2001
SPI assumes cultural changes, so we need expertise from social sciences	Conradi & Fuggetta 2002
Improvement Management	38
General Guidance	*19*
The willingness of management to take risks	Goldenson & Herbsleb 1995
SPI monitoring by senior management	Goldenson & Herbsleb 1995
SPI monitoring by senior management	El Emam et al. 1999
SPI is about learning – not control, as in QA	Conradi & Fuggetta 2002
Understanding of technical issues by senior management	Goldenson & Herbsleb 1995
Be out there. If a quality-concerned group operates by itself, it easily becomes isolated form projects. The quality group needs to spread out and give hands on support to projects. One good way is to select a quality concerned person from the development group and assign him or her to a process concerned person.	Jakobsen 1998
Investment in teamwork operating with the "your success is my success" principle is likely to pay back.	Jakobsen 1998
Combination of technical and methodological aspects	Bazzana & Fagnoni 1999
SPI should be run faster than the development project, thus the complexity and product size will not override the improvement efforts made on project level.	Delmiglio et al. 1999
New technology and tools are often a necessity for successful SPI	Delmiglio et al. 1999

An SPI approach should utilise different improvement approaches along the different SPI initiatives phases	Delmiglio et al. 1999
An SPI method has to be tailorable; “It is a dream to think that the same improvement approach can be applied everywhere, p. 283”.	Debou et al. 1999
From the methodological point of view, in addition to willingness for change, a success factor can be found in wise interpretation of ISO 9001 and CMM	Debou et al. 1999
Mid-managers play a big role in SPI. It is also recommended to dampen any overly positive expectations of the engagement parties; achieving convincing and repeatable results may take some time.	Conradi & Fuggetta 2002
Before starting any huge assessment efforts, the organisation’s readiness for improvement should be ensured. It is argued that the assessment process has been greatly overemphasised and that it should be seen only as a starting point, in their words as “just a tiny part of the iceberg”(p. 282).	Debou et al. 1999
It is critical to define the strategy for proceeding from assessment results to implementation of actions. The experience has shown that if too much time elapses from the assessment to the first impact on projects, the motivation is likely to decrease at all levels.	Debou et al. 1999
Improvement program needs to be guided by an improvement approach or a life cycle. There is no point in the program if there is no plan for managing improvement actions.	McGuinness 1996, 1999
A strong participation of as many of the people involved as possible has to be ensured. Improvement should be carried out “by people to people”. While any imposed solutions are likely to be strongly opposed, serious actions are appreciated by engineers.	McGuinness 1996, 1999
Large organisations have to survey all of the methods and approaches available on the market before piloting or implementing solutions across the company.	McGuinness 1996, 1999
Staffing the SPI Initiative	*13*
SPI people should be well respected	Goldenson & Herbsleb 1995
A self test for analysing what kind of persons should be allocated to different projects tasks is proposed; psychological competence model.	Jakobsen 1998
SPI people should be well respected	El Emam et al. 1999
SPI program has to be staffed by SPI experts, middle managers and affected staff members.	Delmiglio et al. 1999
Clear, compensated SPI assignments	Goldenson & Herbsleb 1995
Compensated SPI responsibilities	El Emam et al. 1999
Involvement of people from different departments	Bazzana & Fagnoni 1999
Ultimately, everyone must be involved. Software Engineering is a team effort, and anyone who does not participate in improvement will miss the benefits and may even inhibit progress	Humphrey 1989, p. 19
Use of improvement teams formed of experienced practitioners	Kauppinen & Kujala 2001
Technical staff involvement in SPI	Goldenson & Herbsleb 1995

Involvement of technical staff in SPI is one of the most critical issues	El Emam et al. 1999
Human resources are the key to success. Tasks should be occupied by the right persons and skills. For example, developers gained early success once they took over their own key practices.	Quinn 1996
It is important to establish clear responsibilities and also to set up a mechanism for implementing changes in practice, in the form of a project.	Quinn 1996
Training	*8*
Change is difficult to accomplish. Mistrust and protectionism can be overcome by open discussion, training and awareness sessions conducted, for example, by an external consultant.	Quinn 1996
Specific, just-in-time training and information enhance team performance.	McGuire 1996
Training program has to be defined, and performed regularly	Delmiglio et al. 1999
Focus on the significance of internal training	Bazzana & Fagnoni 1999
Companies underline the role of training, it needs to planned carefully and have clear objectives.	McGuinness 1996, 1999
SPI related training	Lepasaar et al. 2001
It should be planned how to maintain a continuous progression of the project, by using, e.g., small review sessions several times per week.	Jakobsen 1998
Practical support offered by an internal support group is decisive. Support should be provided as implementation assistance, coaching all the way through the project, in gathering measurement data and analysing it, and in form of assuring management commitment.	Rodenbach et al. 2000
Plan	26
Current State Analysis	8
Effective change requires a goal and knowledge of the current process. To use a map, you must know where you are.	Humphrey 1989
Using audits to start the improvement actions proved to be successful.	Quinn 1996
Developers are motivated for change; if possible, start bottom-up with concrete initiatives	Conradi & Fuggetta 2002
SPI program is best started with assessment performed by an external actor	Delmiglio et al. 1999
The different variations of CMM assessment used by the case companies to initiate SPI programs showed equally good results. While the type of assessment may vary, a clear picture of the starting point is a necessity for SPI.	McGuinness 1996, 1999
It is not recommended to start with a large assessment, but rather to use a simple and focused scorecard.	Conradi & Fuggetta 2002
It is proposed to use multidisciplinary improvement teams to perform empirical studies on how people actually work.	Conradi & Fuggetta 2002
"Common sense is the most valuable tool of a process improvement expert. If you are an expert, you don't need a quantitative assessment to identify major improvement potentials" (Lanzerstorfer et Scherzer 1999, p. 378).	Lanzerstorfer & Scherzer 1999
Goal Definition	11
Link to business goals and active role of business managers are important for success.	Debou et al. 1999

SPI Framework should support improvement strategies focusing on goal-orientation and product innovation	Conradi & Fuggetta 2002
SPI initiatives need to be directed at business goals, while also starting with addressing the most pressing needs of the company or project.	Conradi & Fuggetta 2002
The model used for appraisal is not as important as the role of business goals	Kunzmann-Combelles 1996
SPI goals should be well understood	Goldenson & Herbsleb 1995
Improvement goals have to be realistic and visible, while the improvement steps have to start small, one SPI effort at a time.	Conradi & Fuggetta 2002
Start from the problems, not the solutions.	Jakobsen 1998
The capability to select realistic improvement actions	Kauppinen & Kujala 2001
Setting measurable goals	Kauppinen & Kujala 2001
Measurable targets set for SPI work	Lepasaar et al. 2001
It is important to ensure that SPI goals are well understood	El Emam et al. 1999
Improvement Planning	7
Deployment requires a detailed plan	Delmiglio et al. 1999
The importance of detailed improvement planning is highlighted.	Bazzana & Fagnoni 1999
Detailed improvement plan needs to be established, including tasks, schedules, and resources.	Simon 1999
Improvement does not happen by itself. It has to be planned and tracked as a project with a stretch goal.	McGuinness 1996, 1999
Start small, step by step, "take one bit of the elephant at a time" and wait for benefits before extending the improvement areas.	McGuinness 1996, 1999
Avoid long textual documents, try to visualise as much as possible	Jakobsen 1998
A quality plan or quality requirements need to be set and followed up during the improvement project.	Simon 1999
Do	6
Four companies are deploying the Groupware tools in support of process implementation. Automated support should be provided whenever possible.	McGuinness 1996, 1999
Case studies have to be performed before major changes in process and/or used technologies	Delmiglio et al. 1999
Deployment in two pilot projects	Bazzana & Fagnoni 1999
It is recommended to deploy new guidelines using the bottom-up approach and to do that only with the authorisation of a project leader, only after he has discussed and accepted the new guidelines with involved engineers.	Bazzana & Fagnoni 1999
The managers attention can be improved and the improvement speed accelerated by using external consultants in the improvement take up phase.	Debou et al. 1999
Operational guidelines for software development projects are very useful and provide a general framework for projects to proceed in practice.	Simon 1999

Check	8
Need of assessing effectiveness	Kunzmann-Combelles 1996
Results of collected metrics have to be reported regularly, e.g., monthly	Conradi & Fuggetta 2002
In case the pilot project does not show any benefit, no deployment should take place; quantitative results have to be the major criteria for further actions to be taken.	Debou et al. 1999
A continuous follow-up with metrics is emphasised.	Debou et al. 1999
Special care should be paid when showing improvements to technical staff, who are proud of their work and not willing to change their work procedures for fun	Delmiglio et al. 1999
The cost-benefit analysis requires novel amortisation models	Conradi & Fuggetta 2002
Since it is generally difficult to show the ROI of SPI, the need for developing company-specific cost-benefit models is emphasised.	Conradi & Fuggetta 2002
Metrics help verify the improvements and point out further improvement potentials.	McGuinness 1996, 1999
Act	4
Feedback loop and facilitation are the factors that enable successful process improvement, while also sustaining it.	McGuire 1996
Establishment of continuity in the SPI work.	Lepasaar et al. 2001
Change is continuous. Software process improvement is not a one-shot effort; it involves continual learning and growth.	Humphrey 1989, p. 19
Software process changes will be not retained without conscious effort and periodic reinforcement.	Humphrey 1989, p. 19

Appendix C: Profes User Manual, Table of Contents

Papers I–VII of this publication are not included in the PDF version. Please order the printed version to get the complete publication (http://www.vtt.fi/inf/pdf/)

Published by

Vuorimiehentie 5, P.O.Box 2000, FIN-02044 VTT, Finland
Phone internat. +358 9 4561
Fax +358 9 456 4374

Series title, number and report code of publication

VTT Publications 535
VTT-PUBS-535

Author(s)
Komi-Sirviö, Seija

Title

Development and Evaluation of Software Process Improvement Methods

Abstract

Software development is in constant change. New software development strategies, methods, processes, and tools are constantly introduced and taken in use. Simultaneously, the growth and importance of software has accelerated, and software has become a fundamental part of a whole range of different products. Software development strategies are changing as well: globally distributed software development, use of commercial off-the-shelf (COTS), and Open Source development are some examples of the latest tendencies. Ever-tightening competition has led to shortened lead-time requirements and variety of customised software versions targeted to divergent markets. Software development needs to be optimised to meet these challenges - without sacrificing quality. To keep abreast of change software process improvement (SPI) should develop, too, over time.

Well-managed software development processes has become strategic core competency in many organisations, enabling high-class software development, quality estimation, control, and prediction. However, improving software development processes is demanding and complex task. Numerous software process improvement (SPI) methods in the market offer help and guidance, but unfortunately they only partially address factors found essential for achieving SPI success.

This dissertation develops, presents and argues for the SPI methods embodying characteristics directing towards successful process improvement. As the results, the thesis extracts critical success factors for SPI initiatives using SPI lessons learnt. Furthermore, it incrementally develops and evaluates SPI methods, incorporating means to achieve the above-mentioned critical success factors. The research is based on several industrial case studies.

Keywords
software process improvement (SPI), SPI measurement, software quality, knowledge management, Pr^{2}imer, Profes

Activity unit
VTT Electronics, Embedded Software, Kaitoväylä 1, P.O.Box 1100, FIN-90571 OULU, Finland

ISBN 951–38–6388–3 (soft back ed.) 951–38–6389–1 (URL: http://www.vtt.fi/inf/pdf/)			Project number
Date June 2004	Language English	Pages 175 p. + app. 78 p.	Price
Name of project		Commissioned by	
Series title and ISSN VTT Publications 1235–0621 (soft back ed.) 1455–0849 (URL: http://www.vtt.fi/inf.pdf/)		Sold by VTT Information Service P.O.Box 2000, FIN-02044 VTT, Finland Phone internat. +358 9 456 4404 Fax +358 9 456 4374	

VTT PUBLICATIONS

514 Koskela, Juha. Software configuration management in agile methods. 2003. 54 p.

516 Määttä, Timo. Virtual environments in machinery safety analysis. 2003. 170 p. + app. 16 p.

515 Palviainen, Marko & Laakko, Timo. mPlaton - Browsing and development platform of mobile applications. 2003. 98 p.

517 Forsén, Holger & Tarvainen, Veikko. Sahatavaran jatkojalostuksen asettamat vaatimukset kuivauslaadulle ja eri tuotteille sopivat kuivausmenetelmät. 2003. 69 s. + liitt. 9 s.

518 Lappalainen, Jari T. J. Paperin- ja kartonginvalmistusprosessien mallinnus ja dynaaminen reaaliaikainen simulointi. 2004. 144 s.

519 Pakkala, Daniel. Lightweight distributed service platform for adaptive mobile services. 2004. 145 p. + app. 13 p.

520 Palonen, Hetti. Role of lignin in the enzymatic hydrolysis of lignocellulose. 2004. 80 p. + app. 62 p.

521 Mangs, Johan. On the fire dynamics of vehicles and electrical equipment. 2004. 62 p. + app. 101 p.

522 Jokinen, Tommi. Novel ways of using Nd:YAG laser for welding thick section austenitic stainless steel. 2004. 120 p. + app. 12 p.

523 Soininen, Juha-Pekka. Architecture design methods for application domain-specific integrated computer systems. 2004. 118 p. + app. 51 p.

524 Tolvanen, Merja. Mass balance determination for trace elements at coal-, peat- and bark-fired power plants. 2004. 139 p. + app. 90 p.

525 Mäntyniemi, Annukka, Pikkarainen, Minna & Taulavuori, Anne. A Framework for Off-The-Shelf Software Component Development and Maintenance Processes. 2004. 127 p.

526 Jäälinoja, Juho. Requirements implementation in embedded software development. 2004. 82 p. + app. 7 p.

527 Reiman, Teemu & Oedewald, Pia. Kunnossapidon organisaatiokulttuuri. Tapaustutkimus Olkiluodon ydinvoimalaitoksessa. 2004. 62 s. + liitt. 8 s.

528 Heikkinen, Veli. Tunable laser module for fibre optic communications. 2004. 172 p. + app. 11 p.

529 Aikio, Janne K. Extremely short external cavity (ESEC) laser devices. Wavelength tuning and related optical characteristics. 2004. 162 p.

530 FUSION Yearbook. Association Euratom-Tekes. Annual Report 2003. Ed. by Seppo Karttunen & Karin Rantamäki. 2004. 127 p. + app. 10 p.

531 Toivonen, Aki. Stress corrosion crack growth rate measurement in high temperature water using small precracked bend specimens. 2004. 206 p. + app. 9 p.

532 Moilanen, Pekka. Pneumatic servo-controlled material testing device capable of operating at high temperature water and irradiation conditions. 2004. 154 p.

535 Komi-Sirviö, Seija. Development and Evaluation of Software Process Improvement Methods. 2004. 175 p. + app. 78 p.

Tätä julkaisua myy	Denna publikation säljs av	This publication is available from
VTT TIETOPALVELU	VTT INFORMATIONSTJÄNST	VTT INFORMATION SERVICE
PL 2000	PB 2000	P.O.Box 2000
02044 VTT	02044 VTT	FIN–02044 VTT, Finland
Puh. (09) 456 4404	Tel. (09) 456 4404	Phone internat. +358 9 456 4404
Faksi (09) 456 4374	Fax (09) 456 4374	Fax +358 9 456 4374

ISBN 951–38–6388–3 (soft back ed.) ISBN 951–38–6389–1 (URL: http://www.vtt.fi/inf/pdf/)
ISSN 1235–0621 (soft back ed.) ISSN 1455–0849 (URL: http://www.vtt.fi/inf/pdf/)