V T T P U B L I C A T I O N S

Janne Järvinen

Measurement based continuous assessment of software engineering processes

TECHNICAL RESEARCH CENTRE OF FINLAND ESPOO 2000

VTT PUBLICATIONS 426

Measurement based continuous assessment of software engineering processes

Janne Järvinen

VTT Electronics

Academic Dissertation to be presented with the assent of the Faculty of Science, University of Oulu, for public discussion in Auditorium L6, on December 15th, 2000, at 12 noon.

TECHNICAL RESEARCH CENTRE OF FINLAND
ESPOO 2000

ISBN 951–38–5592–9 (soft back ed.)
ISSN 1235–0621 (soft back ed.)

ISBN 951–38–5593–7 (URL: http://www.inf.vtt.fi/pdf/)
ISSN 1455–0849 (URL: http://www.inf.vtt.fi/pdf/)

Copyright © Valtion teknillinen tutkimuskeskus (VTT) 2000

JULKAISIJA – UTGIVARE – PUBLISHER

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

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

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

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

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

VTT Electronics, Embedded Software, 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 2000

In theory, there is no difference between theory and practice; In practice, there is.

-- Chuck Reid

Järvinen, Janne. Measurement based continuous assessment of software engineering processes. Espoo 2000, Technical Research Centre of Finland, VTT Publications 426. 97 p. + app. 90 p.

Keywords software engineering, software process, software process assessment, software measurement, software process improvement

Abstract

Software process assessments are routinely used by the software industry to evaluate software processes before instigating improvement actions. They are also used to assess the capability of an organisation to produce software. Since assessments are perceived as expensive, time-consuming and disruptive for the workplace there is a need to find alternative practices for software process assessment. Especially interesting for this research was to understand and improve the way an organisation can monitor the software process status between regular assessments and how this monitoring can be achieved feasibly in an industrial setting.

This thesis proposes a complementary paradigm for software process assessment – measurement based continuous assessment. This approach combines goal-oriented measurement and an emerging standard for software process assessment as the background framework for continuous assessment of the software engineering process. Software tools have been created to support the approach that has been tested in an industrial setting. The results show that the proposed approach is feasible and useful, and provides new possibilities and insights for software process assessment.

Preface

This research was carried out during 1996 to 2000 at VTT Electronics, Oulu, in the PROAM and PROFES projects and at Fraunhofer IESE, Germany, in the FAME project. The basis for this research was laid in the VTT Electronics' strategic project PROAM and the SPICE project during 1996. Special thanks to Mary Campbell for her ideas and encouragement to explore continuous assessment. The major part of this research was done during the European ESPRIT project #23232 PROFES where the continuous assessment concepts were strengthened and tested in an industrial environment. The FAME project provided a chance to document the assessment types and modes found in practice. My sincere thanks to Dr. Veikko Seppänen, Dr. Markku Oivo, Prof. Dr. Dieter Rombach and Prof. Dr. Günther Ruhe for giving me opportunity to work in these exciting projects. This research has financially been also supported by Finnish Cultural Foundation and Anja and Jalo Paananen Foundation.

I would like to thank my supervisor at the University of Oulu, Dr. Jouni Similä, for guiding my research efforts for the past ten years. I am most grateful to Dr. Helen Thomson and Dr. Antti Auer for spending their time and providing constructive critique as the nominated reviewers of this thesis. Special thanks to Dr. Veikko Seppänen for always being available to read my incomplete works.

I want to thank all my colleagues at VTT Electronics and co-operation partners of other organisations. Especially those who have co-authored papers with me for this thesis deserve credit for their patience: Andrew Beitz, Adriana Bicego, Andreas Birk, Tomi Dahlberg, Dirk Hamann, Seija Komi-Sirviö, Pasi Kuvaja, Päivi Parviainen, Dietmar Pfahl, Toni Sandelin, Rini van Solingen and Matias Vierimaa. I am also grateful to Erik Rodenbach, Pieter Derks, Rob Koll, Arnim van Uijtregt and especially Niels van Veldhoven for co-operating with me during the case studies.

Finally, I wish to express my warmest gratitude to my parents Sirkka and Olli, and my family: Kicka, Kalle and Saku, for their support and patience during these trying times.

Lannevesi, September 2000 Janne Järvinen

List of original publications

This dissertation includes the following eight original publications:

I Dahlberg, T. & Järvinen, J. 1997. Challenges to IS Quality. Information and Software Technology Journal, Vol. 39, No.12, pp. 809 - 818.

II Parviainen, P., Järvinen, J. & Sandelin, T. 1997. Practical Experiences of Tool Support in a GQM-based Measurement Programme. Software Quality Journal, Vol. 6, No. 4, pp. 283 - 294.

III Järvinen, J. 1998. Facilitating Process Assessment with Tool Supported Measurement Programme. In: Coombes, H., Hooft van Huysduynen, M. & Peeters, B. (eds.), Proceedings of FESMA98 – Business Improvement Through Software Measurement. Technological Institute, Antwerp, Belgium, May 6 - 8, pp. 606 - 614.

IV Birk, A., van Solingen, R., & Järvinen, J. 1998. Business Impact, Benefit, and Cost of Applying GQM in Industry: An In-Depth, Long-Term Investigation at Schlumberger RPS. In: Proceedings of the Fifth International Symposium on Software Metrics (METRICS´98). Bethesda, Maryland, November 20 - 21, pp. 93 - 96.

V Järvinen, J. & van Solingen, R. 1999. Establishing Continuous Assessment Using Measurements. In: Proceedings of the 1st International Conference on Product Focused Software Process Improvement (PROFES´99). Oulu, Finland, June 22 - 24, pp. 49 - 67.

VI Vierimaa, M., Hamann, D., Komi-Sirviö, S., Birk, A., Järvinen, J. & Kuvaja, P. 1999. Integrated Use of Software Assessments and Measurements. In: Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering (SEKE '99). Kaiserslautern, Germany, June 17 - 19, pp. 83 - 87.

VII Hamann, D., Pfahl, D., Järvinen, J. & van Solingen, R. 1999. The Role of GQM in the PROFES Improvement Methodology. In: Proceedings of the 3rd International Conference on Quality Engineering in Software Technology (CONQUEST '99). Nürnberg, Germany, September 26 - 27, pp. 64 - 79.

VIII Järvinen, J., Hamann, D. & van Solingen, R. 1999. On Integrating Assessment and Measurement: Towards Continuous Assessment of Software Engineering Processes. In: Proceedings of the Sixth International Symposium on Software Metrics (METRICS´99). Boca Raton, Florida, November 4 - 6, pp. 22 - 30.

Contents

Appendices of this publication are not included in the PDF version. Please order the printed version to get the complete publication (http://otatrip.hut.fi/vtt/jure/index.html)

List of Names and Acronyms

AMI	A quantitative process improvement paradigm
BOOTSTRAP	European software process assessment and improvement methodology (http://www.bootstrap-institute.com/)
CBA-IPI	CMM-Based Appraisal for Internal Process Improvement
CMM	Capability Maturity Model (http://www.sei.cmu.edu/cmm)
CMMI	Capability Maturity Model Integrated
CMM-SW	Capability Maturity Model for software
CAF	CMM Appraisal Framework
EFQM	European Foundation for Quality Management (http://www.efqm.org)
FAME	Fraunhofer IESE Assessment Method
GQM	Goal-Question-Metric method (e.g. http://www.gqm.nl)
IEC	International Electrotechnical Commission (http://www.iec.ch/home-e.htm)
IEEE	Institute of Electrical and Electronics Engineers (http://www.ieee.org)
IS	Information Systems
ISO	International Standardization Organisation (http://www.iso.ch)
IT	Information Technology
KLOC	Kilo Lines of Code
KPA	Key Process Area
MAA	A preliminary version of MCA
MCA	Measurement-based Continuous Assessment
OPT	Outdoor Payment Terminal
PAMPA	A software visualisation toolkit
POS	Point-Of-Sales
PPD	Product-Process Dependency
Pr^{2}imer	Practical Process Improvement for Embedded Real-Time Software
PROFES	Product Focused improvement methodolology (http://www.profes.org)
PSP	Personal Software Process
QIP	Quality Improvement Paradigm
QPR	Quality Problem Report
RPM	A conceptual model for product focused SPI
SCE	Software Capability Evaluation
SEI	Software Engineering Institute

SME	Small and Medium-size Enterprise
SPA	Software Process Assessment
SPI	Software Process Improvement
SPICE	Software Process Improvement and Capability dEtermination (e.g. http://www.iese.fhg.de/SPICE)
TSP	Team Software Process
TEKES	National Technology Agency of Finland (http://www.tekes.fi)
TQM	Total Quality Management
VCS	Version Control System
VTT	Technical Research Centre of Finland (http://www.vtt.fi)
WP	Work Product

1. Introduction

1.1 Background

Software has become more and more pervasive in our society. The year 2000 bug is a good example of the grip software has on the global economy. As the need for improved functionality, quality, and reliability continues, better ways to control software development are needed. Also, as Drouin (1999, p. 45) writes, ”aside from being a major cause of disappointment and frustration, software failures have become a major source of financial drain on organisations at a time when cost containment is the chief concern of senior managers”.

In the 1990´s the software process community grew with the importance of software in the industry (Humphrey 1999). The paradigm of the software process proponents is that the quality of software development process is in close relationship with the quality of the resulting software (Humphrey 1989, p. 13). There are good examples and evidence that an improved software development process also increases productivity and reduces variation of the software development process (Humphrey et al. 1991; Clark 1997; Herbsleb et al. 1997). Krasner (1999, p. 151) writes: “In a mature software organisation, the following holds:

- Quality is defined and therefore predictable
- Costs and schedules are predictable and normally met
- Processes are defined and under statistical control”.

Software process improvement (SPI) is not cheap or necessarily easy to implement. Jones (1999, p. 133) writes that “the cost per capita for major software process improvements can exceed $25,000 and the timing can exceed five years in large corporations”. For example, The Raytheon Software Systems Laboratory in the Equipment Division had the goal of transitioning from CMM-SW Maturity Level 1 to Level 3 (Dion 1993). This initiative took five years and the division invested almost $1 million. Hence, it is not surprising that software process improvement is sometimes seen mostly oriented towards large and highly structured organisations (Cugola & Ghezzi 1998). However, many organisations who have implemented improvement processes can quantify the high return for their investment. Raytheon, for example, reported a $7.7 return for every $1 invested (Dion 1993). A study of 13 organisations (Herbsleb et al. 1994) lists also other benefits from SPI (Table 1). On the other hand, there are problems with SPI in the industry. Rombach (2000) states that "software process improvement activities in industry often fail producing sustained improvements" and Krasner (1999) claims that two-thirds of formal SPI programs die soon after a formal assessment.

Table 1. Examples of SPI benefits (Herbsleb et al. 1994, p. 15).

Category	**Range**	**Median**
Return of Investment	4.0 - 8.8	5.0
Productivity gain per year	9% - 67%	35 %
Reduction in time to market	15% - 23%	19%
Pre-test defect detection gain per year	6% - 25%	22%
Yearly reduction in post-release defects	10% - 94%	39%

According to the Software Engineering Institute (SEI), the number of organisations initiating SPI continues to increase, and nearly half of the organisations reporting size for the SEI database have a software personnel of 100 people, or less (SEMA 2000). The SPICE trials report organisational units between 10 and 500 IT personnel from different business sectors participating in trialing a software process assessment standard (SPICE 1998). However, even as the interest broadens and experiences from SPI grow, little has been reported from the development of software assessment practice. The team-based assessment remains as the prevalent means of assessment, although the need for alternative practices for software process assessment has been acknowledged (Campbell 1995; Miyazaki et al. 1995).

1.2 Scope of the research

According to Curtis et al. (1995, p. 10), there are three different success criteria in software engineering: people, process and technology. This research deals mainly with the process aspects of software engineering. Within process oriented research there are four main approaches for software process improvement (Kuvaja et al. 1994, p. 29): assessment (e.g. Humphrey 1989), modelling (e.g. Kellner & Hansen 1988), measurement (e.g. Basili et al. 1994b) and technology transfer (e.g. Pfleeger 1998) that can be used independently or together. Of these four, process modelling and technology transfer are not part of this research although they have a place within the improvement work. For example, process modelling is needed to analyse existing processes – descriptive process modelling – or to mandate new processes to be used – prescriptive process modelling (Curtis et al. 1992; Bandinelli et al. 1995). It is the integrated use of assessment and measurement, which forms the general scope of this thesis. More specifically, this research involves practical approaches that will support assessment with measurement data. It would seem easy to come up with requirements for a state-of-the-art integrated process support environment that would also provide information for assessment purposes. In practice, it is not easy to find organisations that are using integrated process support environments. The choice in this research was to acknowledge the state-of-the-practice in industrial software development, and focus on how to find methods, techniques and tools for improving assessments that would be useful for the people working in the industry today. This thesis mainly targets organisations of ISO 15504 levels 1 - 3 in maturity. However, similar techniques are apparently being applied in high maturity organisations although their experiences are less often made public. More information on high maturity organisations is being made available as the recent survey by Paulk et al. (2000) exhibits.

This research focuses on the areas related directly to integrating software process assessment and software measurement. Software process improvement approaches are interfaced from this thesis' viewpoint. Other areas, such as process modelling (Kellner & Hansen 1988), process simulation (Abdel-Hamid & Madnick 1991), process automation (Christie 1994), knowledge management (Nonaka & Takeuchi 1995; Davenport et al. 1996) or quality, TQM, statistics, organisational change and people (Crosby 1979; Gryna & Juran 1980; Ishikawa 1985; Deming 1986; Lillrank 1990; Kenett & Zacks 1998; DeMarco & Lister

1999) or Balanced Scorecard (Kaplan & Norton 1996), are among related issues but are not included as they do not fit into the focus area of this thesis.

Finally, this thesis deals mainly with the technical aspects of supporting software process assessments with measurement as this field of research is still at its infancy. In parallel to IS quality dimensions (Eriksson & Törn 1991; Braa 1995) the main interest of this research is to understand the co-use of assessment and measurement in the sense of technical quality (Figure 1). There has been some attempt to evaluate the satisfaction and effectiveness of the proposed approach but more empirical evidence and multi-perspective research are needed before these dimensions are adequately covered.

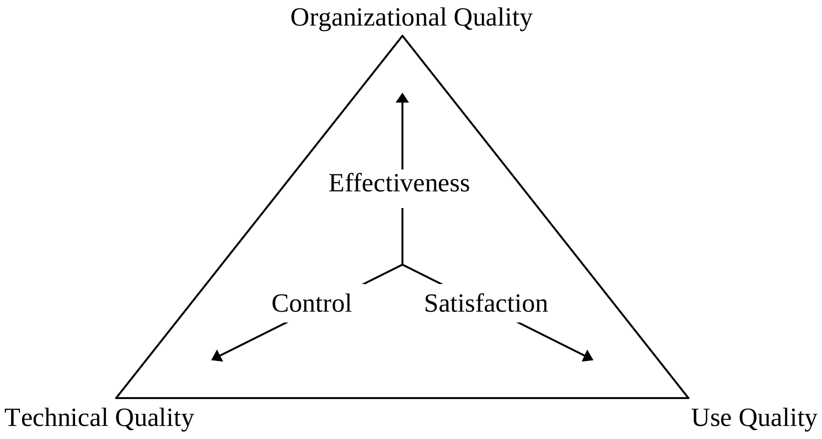


Figure 1. The IS quality dimensions (Eriksson & Törn 1991; Braa 1995).

1.3 Research problem

This research is motivated by the problems with software process assessments, which are reported in the literature and in practice. Although assessments are generally considered useful, there are aspects in some of the current approaches that make assessments perceived to be too expensive, disruptive, infrequent or inflexible (Bollinger & McGowan 1991; Barker et al. 1992; Card 1992; Campbell 1995; Drouin 1999; Johnson & Brodman 1999). In addition, there is little reported on the development of software assessment practice, although the interest for assessments is on the rise (SEMA 2000), and there is expressed interest for alternative practices for software process assessment (Campbell 1995; Miyazaki et al. 1995). An assumption of this research is that some of the problems with software process improvement are caused by the shortcomings of today's assessment methods. For example, it is easy to go astray with a process improvement program when the process capability status is assessed only every other year. Another assumption of this research is that better monitoring of the status of the software process helps to control improvement activities, thus reducing reported problems of discontinuity (Kinnula 1999; van Solingen 2000). The third assumption is that although software measurement seems to be a relatively established field where methods and techniques for measuring software products and processes are commonplace, what seems to be lacking is the use of reference frameworks to better understand, manage and utilise measurements. The research problem can then be formulated as follows:

- How can an industrial organisation monitor the status of its software process using measurement based continuous assessments?

Based on the research problem the following research questions arise:

- How does continuous assessment differ from other assessments?
- What techniques and support are needed for establishing measurement based continuous assessment?
- Is it feasible to use measurement based continuous assessment in an industrial setting?

In summary, this thesis asserts the following hypothesis:

- Measurements may successfully be embedded into the software process to support regular process assessment.

Especially when people are skilled and empowered, analysing measurement data against an assessment framework yields extra benefits for understanding, controlling and improving the knowledge intensive software work.

1.4 Research setting

1.4.1 Research approach

This research may be characterised as applied research in the field of software engineering. More specifically the research approach was constructive research (Järvinen 1999, p. 59) that includes conceptual development, technical solutions, and their evaluation. Glass (1994) calls this approach “the engineering method”. Trochim (1997) agrees with this constructivist approach and claims it represents the post-positivist thinking in contemporary science.

The intention of this research was to understand software process assessment and to construct automated and integrated support for doing the assessment as frequently as needed. An a priori assumption is that there is a need and added value in finding new approaches and support for software process assessment. This assumption was an internal quality criterion for this research and it was considered at various stages of this work.

1.4.2 Research methods

The research field is changing very rapidly as the software process community is constructing best practices and frameworks for software processes and their assessment. The development of the recent ISO standards ISO 12207 and ISO 15504 is a good example of this. The ISO 12207 (1995) describes the practices needed to fulfil the requirements of the standards – the best practices. However, some of these practices are considered applicable only in a specific situation from the viewpoint of the emerging standard ISO 15504 (1998). Then, there is the effort to collect the body of knowledge in software engineering (Bourque et al. 2000) that seems not to be fully consistent with ISO 12207 and ISO 15504. Therefore, this research is constructive – attempting to find solutions for problems in this emerging field. Järvinen (1999, p. 61) notes that this approach relates to the technology oriented design science asking “Can we build a construct, model, method or instantiation to be utilised?”. In practice, the research has been an interplay between conceptual sense making, construction of methods, techniques and tools, and empirical evaluation of the results.

Conceptual analysis

The state-of-the-art of software process assessment has been analysed using both literature and active involvement in the field as an assessor and in the development of ISO 15504, the emerging standard for software process assessment (1998). The recognised problems have led to the characterisation of challenges for IS quality (Paper I) and the typology of the different assessment types and modes. Greater understanding of industrial software measurement (Paper IV) helped to see new possibilities and the initial concept of measurement based continuous assessment (MCA) (Paper III). MCA has been refined in the PROFES project (Papers V, VIII) and integrated to the PROFES improvement methodology (Papers VI, VII) and (PROFES-Consortium 2000).

Constructive tasks

A method for applying measurement based continuous assessment (MCA) is developed (Papers V, VIII). Tools for measurement data management (VTT 1999), mapping measurement data to reference framework (VTT 2000) and assessment profile monitoring (Etnoteam 1998) as well as several templates have been built to support the proposed method (PROFES-Consortium 2000). Furthermore, the MCA method is integrated into the PROFES improvement methodology (Papers VI, VII) and (PROFES-Consortium 2000), and the FAME assessment methodology (Beitz & Järvinen 2000).

Empirical studies

Case studies have been used to evaluate the MCA approach (Papers V, VIII). Case studies are most useful to perform research to answer “why” and “how” questions, which do not require control over behavioural events, and which focus on contemporary events (Yin 1991, p. 17). The tentative solution for MCA has been reviewed several times in the PROFES project by both academics and practitioners (PROFES-Internal 1997 - 1999). MCA has also been used in an industrial setting for discovering knowledge (Paper V) and for validation of the proposed approach (Paper VIII). This has included the use, evaluation and modification of the method for MCA against the criteria of value according to March and Smith (1995) questioning “does it work, is it an improvement?”

1.4.3 Research process and limitations

This research covers work between 1996 and 2000 in the area of software process assessment and improvement. Early initial ideas for the dissertation came in 1994 - 1996 when the author started to be active in Bootstrap and CMM assessments and their comparison (Järvinen 1994b), being involved in the software process assessment standardisation work as a core member of the assessment instrument team in the SPICE project and the product manager for the Bootstrap assessment tools (Järvinen 1994a). Mary Campbell, the assessment instrument team leader in the SPICE project has summarised many of the needs for alternative practices in (Campbell 1995). At that time the interest was more focused on studying the extent of automatisation of assessment and related tool support. Later, both of these areas became of less importance to the focus of this thesis, but perhaps they would have had more significance if SPICE had remained a closed, tightly focused standard as was originally proposed (Dorling & Simms 1992; SPICE-5 1995).

The author led the VTT Electronics' strategic project PROAM in 1996 - 1997 where assessment automatisation and tool support was studied and the first versions of the MetriFlame tool were made. Experiences of supporting a measurement program with MetriFlame are recorded in Paper II. Results include a preliminary classification of the suitability of SPICE processes for automation (Parviainen et al. 1996). Other deliverables, working documents, interview notes and experimental tools also exist from the PROAM project (PROAM 1996). In 1997, a summary of the literature relating to quality in information technology was made along with discussion on future challenges to quality (Paper I). The PROFES project began in 1997, but in terms of this thesis a more interesting period began early 1998 when the initial concepts for the measurement based continuous assessment (MCA) were laid (Paper III), although precise naming of the concepts did not stabilise until late 1999.

In the PROFES project the author formed a close relationship as a methodology coach with Tokheim in Bladel, the Netherlands (earlier Schlumberger RPS), where he conducted two Bootstrap process assessments in 1997 and 1998 (PROFES-Internal 1997 - 1999) and became aware of the company's improvement and measurement programs. A summary of the experiences from the

Tokheim measurement programs (then still Schlumberger RPS) is recorded in Paper IV.

The concepts for MCA were further developed and integrated to the PROFES improvement methodology in late 1998 and 1999 with active participation and extensive reviewing by the members of the PROFES consortium (PROFES-Internal 1997 - 1999). These developments are presented in Papers V, VI and VII. In addition, a study of MCA was carried out at Tokheim in co-operation with their personnel that included a student who made his masters thesis on continuous assessment (van Veldhoven 1999). The results of the Tokheim MCA study are summarised in Paper VIII. Another study of the MCA was made with Dräger Electronics in Best, the Netherlands (PROFES-Internal 1997 - 1999). The MCA research with Tokheim and Dräger has been recorded in a cost model and several interview notes, working documents and personal emails (PROFES-Internal 1997 - 1999). Also a third case for MCA was initially planned but later rejected due to lack of resources.

Meanwhile, the author participated in the development of the Fraunhofer IESE Assessment method (FAME) as he stayed in Kaiserslautern between 1998 - 1999 as a visiting researcher at the Fraunhofer IESE. For the interest of this thesis the concepts for assessment types and modes discussed in Chapter 3 and the MCA approach are integrated into the FAME approach (Beitz et al. 1999; Beitz & Järvinen 2000).

There are a number of limitations and biases inherent in this study. Firstly, what is remarkable about research within the software process community is that the research is seldom based on solid theory. Instead, most prevalent approaches and standards, such as CMM and ISO 15504, are based on collections of heuristics and industry best practice. The same applies for this research. The MCA method was built based on practical experiences from the limited industry sample, which characterises the applied nature of this research and the early stage of the maturity of the research in software process community. Secondly, another limitation of this study was the limited number of cases. Due to lack of resources from both the researchers and industrial partners, the validation of the approach has been based on two case studies. However, the extensive reviewing of the concepts and operational definition of MCA during the PROFES project was helpful for maturing of the method. Hence, better understanding and knowledge on the relationships between assessment and measurement were acquired during the research.

1.5 Outline of the thesis

The structure of the thesis is as follows:

- Introduction (Chapter 1) – explains the motivation behind this research, introduces the research problem, the research methods, and explains the focus of the thesis.

- Related work (Chapter 2) – discusses the research relevant to this study.

- Assessment types and modes (Chapter 3) – introduces the different variations of software process assessment to set the stage for this research.

- Measurement based continuous assessment (Chapter 4) – explains the concepts and model of measurement based continuous assessment (MCA). A method for MCA is also presented.

- Case: Applying continuous assessment in an industrial setting (Chapter 5) – describes an application of MCA at Tokheim, Bladel, the Netherlands.

- Summary and Conclusions (Chapter 6) – sums up this research and answers research questions. Finally, directions for further research are presented.

- Introduction to papers (Chapter 7) – explains the original papers that form the basis of this dissertation. The papers are included as appendices in this thesis.

2. Related work

2.1 Software process assessment approaches

2.1.1 Introduction

In the 1990´s the software process community grew with the importance of software in the industry (Humphrey 1999). Many standards included references to software and new software related standards were created to help control software quality and production (DOD-STD-2167A 1988; ISO/IEC-9126 1991; ISO/IEC-12207 1995; ISO/IEC-9000-3 1997; ISO/IEC-15504-2 1998). The standardisation efforts were not always co-ordinated, and even today there remains some confusion as to how the different standards and approaches fit to the big picture. An incomplete and partly inaccurate but illustrative web of the software standards and frameworks relationships also known as the frameworks quagmire (Sheard 1997) is presented in Figure 2.

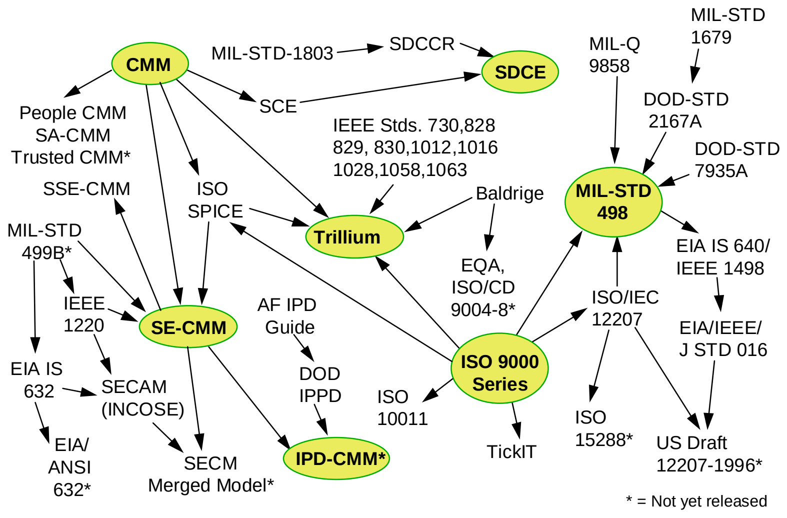


Figure 2. Frameworks quagmire (Sheard 1997).

For the interest of this thesis, the following three software process assessment (SPA) approaches are described: Capability Maturity Model for software (CMM-SW), ISO 15504 (also known as SPICE), the emerging international SPA standard and BOOTSTRAP, an ISO 15504 compliant method for SPA.

There are many other candidates that could have been presented as can be seen from the Figure 2, and the world is changing all the time to produce new standards and approaches. There are qualities in the selected related approaches, however, that make them most suitable candidates from the viewpoint of this research. Firstly, the CMM-SW from the Software Engineering Institute is the pioneer and most well known model for software process capability (Dutta et al. 1996). Secondly, ISO 15504 is an emerging international standard for software process assessment that is expected to be a general reference framework in software process assessment (Drouin 1999). Therefore, ISO 15504 is the primary reference model used in this research for process capability. Thirdly, BOOTSTRAP is the first (and only) ISO 15504 compliant method that was available at the time of the research.

2.1.2 CMM-SW

The Software Engineering Institute's Capability Maturity Models (CMMs) have been developed in recent years for various purposes and there is an integration effort to harmonise all CMMs under one framework - the CMMI (2000). However, CMMI has received criticism (Pierce 2000) for example for being too large and complex (437 practices, 8 KPAs at Level 2 and 11 KPAs at Level 3). The official CMMI was not available at the time of this research. Hence, only the existing official software CMM, the CMM-SW, was chosen for further examination.

Based on the original ideas of Radice et al. (1985) and Humphrey (1989), the CMM-SW v.1.1 has five predefined levels of process capability and a set of key process areas associated with each level describing an evolutionary path from an ad hoc, immature software process (level 1) to a mature, disciplined and optimised software process (level 5) (Paulk et al. 1993). The CMM-SW covers aspects of planning, engineering, and managing software development and maintenance.

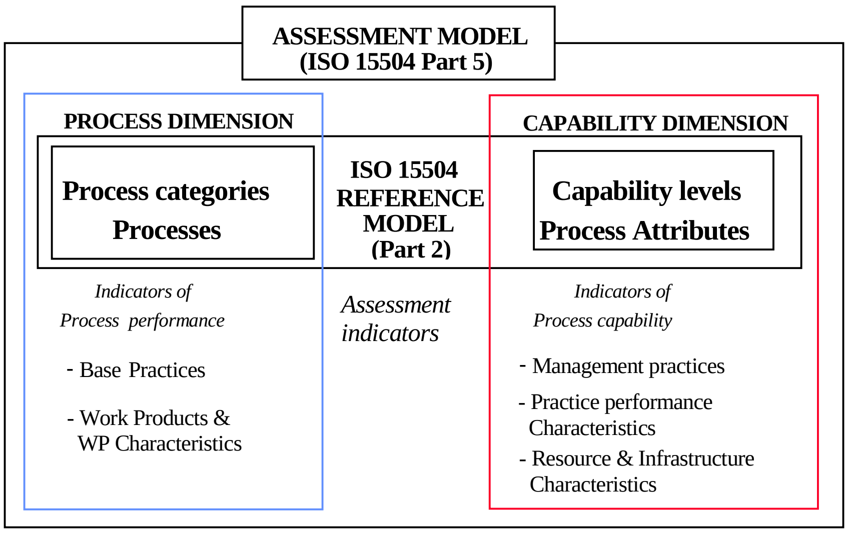
A CMM assessment is a highly structured, team-based activity with on-site document reviews and interviews that are done either for process improvement or capability determination purposes (Masters & Bothwell 1995; Byrnes & Phillips 1996; Dunaway & Masters 1996). Between CMM assessments interim profiles can be made to monitor process capability informally (Whitney et al. 1994). Measurement is considered to be a part of achieving each key process area, and at levels 4 and 5, measurement becomes instrumental for achieving higher capability through quantitative understanding, controlling and optimising of software processes.

2.1.3 ISO 15504

The SPICE project was organised to harmonise efforts for software process assessment after an ISO study report (Dorling & Simms 1992) concluded that there is a need to facilitate the repeatability and comparability of assessment results. The result is ISO 15504 – a framework for software process assessment, which is currently undergoing a trial period before it is considered to be published as an international standard. Recent developments show that ISO 15504 may be integrated more tightly with other related standards (Nevalainen 2000), which may affect the shape and form of the final international standard. However, this research is based on the baselined documents available at the time of writing.

The reference model in Part 2 of the proposed standard (ISO/IEC-15504-2 1998) contains a process dimension with best-practice definitions of software processes and a capability dimension with six levels of process capability. Figure 3 shows the ISO 15504 reference model embedded within the exemplar assessment model ISO 15504 Part 5 (1998). The reference model forms an interface for models and methods to be used for software process assessment. Any model or method wishing to show compliance to ISO 15504 must produce a mapping against the ISO 15504 reference model.

The Part 5 of ISO 15504 (1998) contains an exemplar assessment model with assessment indicators for process performance and process capability. The assessment indicators provide a detailed view of the processes that can be linked to measurements. Assessment indicators contain indicators of process performance and process capability. Process performance indicators are the base practices, i.e. software engineering or management activities that address the purpose of a particular process, and associated work products that have specific characteristics.



ASSESSMENT MODEL (ISO 15504 Part 5)
PROCESS DIMENSION
Process categories
Processes
ISO 15504 REFERENCE MODEL (Part 2)
CAPABILITY DIMENSION
Capability levels
Process Attributes
Indicators of Process performance
- Base Practices
- Work Products & WP Characteristics
Assessment indicators
Indicators of Process capability
- Management practices
- Practice performance Characteristics
- Resource & Infrastructure Characteristics

Figure 3. The ISO 15504 framework for software process assessment (1998).

Process capability indicators are the management practices, i.e. management activities or tasks that address the implementation or institutionalisation of a specified process attribute. Management practices are linked with attribute indicators sets which are: a) practice performance characteristics that provide guidance on the implementation of the practice; b) resource and infrastructure characteristics that provide mechanisms for assisting in the management of the process; and c) associated processes from the process dimension that support the management practice. (ISO/IEC-15504-5 1998, pp. 3 - 4)

2.1.4 Bootstrap

The BOOTSTRAP methodology is an ISO 15504 compliant European methodology for software process assessment and improvement maintained by the Bootstrap Institute (Kuvaja et al. 1994). BOOTSTRAP includes the following components:

- reference model against which the process capability is evaluated,
- assessment method that defines the assessment procedure,
- improvement method that includes guidance on how the assessment results are used for process improvement.

The BOOTSTRAP reference model has two dimensions: a process dimension and a capability dimension. In the assessment, the process dimension is used for identifying the objects for evaluation, and the capability dimension provides the evaluation criteria. The process dimension of the BOOTSTRAP methodology includes processes that fulfil the requirements of ISO standards 9001 (1994), 9000-3 (1997), 12207 (1995) and 15504 (1998), the European Space Agency standard PSS-05-0 (ESA 1991) and the Capability Maturity Model for software v.1.1 (Paulk et al. 1993). The capability dimension of the BOOTSTRAP reference model is aligned with the capability dimension of ISO 15504, including six levels of capability. In addition to output profiles conforming to ISO 15504, the BOOTSTRAP method also generates synthetic profiles using quartiles within the capability levels (Bicego et al. 1998). The BOOTSTRAP methodology was integrated into the PROFES improvement methodology (PROFES-Consortium 2000). Paper VI includes detailed discussion on the integrated use of software process assessments and measurements.

2.2 Software measurement

2.2.1 Software measurement concepts

Measurement is the use of metrics to assign a value from the measurement scale to an attribute or entity (ISO/IEC-8402 1994). Entities are the objects of interest for measurement and attributes are the properties of the entity. The measurement scale type defines the characteristics (Table 2) of suitable measures and analysis techniques (Kitchenham 1996).

Software measurement is the continuous process of defining, collecting, and analysing data on the software development process and its products to understand, control and optimise the process and its products (Fenton & Pfleeger 1996, pp. 14 - 15).

Measurement in software engineering is different from measurement in other engineering disciples and industries. A classic quote from Tom DeMarco (1982) says, “You can’t control what you can’t measure”. Kitchenham (1996, p. 4) paraphrases DeMarco and takes a more critical view on software measurement saying, “Just because you can measure something, it doesn’t mean you can control it”. Brooks (1987) was even more pessimistic by saying that software is invisible and unvisualisable. Hsia (1996) agrees in principle and states that it is very hard to monitor the construction of something you can’t see. However, measurements can be used to make the software development process more visible, and as Mosemann (1994) says, “Software can be engineered, software development can be managed”. There is even some general tool support available for visualising software development, such as the PAMPA toolkit (Simmons et al. 1998). Kitchenham (1996, p. 4) still remains critical and argues that with software and process improvement it is better to use a medical rather than engineering analogy:

“When software practitioners are involved in process improvement, they are rather like a doctor attempting to heal a sick patient. The doctor may need to treat their patient’s immediate symptoms, but they will need to make some diagnosis of the underlying disease if they are to administer an effective treatment…Using this analogy, measurement can be viewed as one of the tests a software practitioner can apply to attempt to diagnose the underlying problem in a software process, and as one of the means of monitoring the response of the software process to the applied treatment (i.e. process change”.

The medical analogy reminds us that software measurement data needs to be analysed by the people who know the data well enough to make reasoned conclusions.

Table 2. Measurement scale types (Kitchenham 1996, p. 61).

Name	Definition	Examples	Constraints
Nominal	A set of categories into which an item is classified.	Testing methods: design inspections, unit testing, integration testing, system testing. Fault types: interface, I/O, computation, control flow.	Categories cannot be used in formulas even if you map you categories to the integers. You can use the mode and percentiles to describe nominal datasets
Ordinal	An ordered set of categories.	Ordinal scales are often used for adjustment factors in cost models based on a fixed set of scale points such as very high, high, average, low, very low. The SEI Capability Maturity Model (CMM) classifies development on a five-point ordinal scale.	Scale points cannot be used in formulas: so 2.5 on the SEI CMM scale is not meaningful You can use medians and percentiles to describe ordinal datasets.
Interval	Numerical values where the difference between each consecutive pair of numbers is an equivalent amount but there is no 'real' zero value. On an interval scale 2 - 1 4 - 3 but 2 units are not twice as much as 1 unit.	lf you have been recording information at six-monthly intervals since 1980, you can measure time since the start of the measurement program on an interval scale starting with 01/01/1980 as 0, followed by 01/06/80 as 1, and 01/01/81 as 2, etc. Degrees Fahrenheit and Celsius are interval scale measures of temperature.	You can use the mean and standard deviation to describe interval scale datasets.
Ratio	Similar to interval scale measures but including an absolute zero.	The number of lines of code in a program is a ratio scale measure of code length. Degrees Kelvin is a ratio scale measure of temperature.	You can use the mean and standard deviation to describe ratio scale datasets.

2.2.2 Software measurement in practice – GQM

Software measurement also needs to be planned and carried out systematically (Grady & Caswell 1987). A measurement program is more likely to succeed if it is based on organisational and project goals (Basili & Rombach 1988). Goal-Question-Metric (GQM) is a goal-oriented approach for setting up and running measurement programs that also addresses the special nature of software measurement. It has been chosen for this research for this reason and because most other goal driven approaches for software measurement in use today (Bassman et al. 1994; Park et al. 1996; Pulford et al. 1996) seem to be adapted from GQM. The approach is also routinely mentioned in books on software measurement as an example of suggested approach for software measurement (Kitchenham 1996; van Solingen & Berghout 1999). There are more details and discussion on GQM and its role as part of the PROFES improvement methodology in Paper VII.

GQM is a well-known and widely used approach for defining and executing goal driven measurement programs (e.g. Basili & Weiss 1984; Basili et al. 1994b; Birk et al. 1998; van Latum et al. 1998). In the GQM approach, high-level goals are used to select measurement goals, which are further refined into questions and metrics that provide information to answer the questions. The GQM approach contains mechanisms for top-down metrics definition and bottom-up data interpretation (Figure 4).

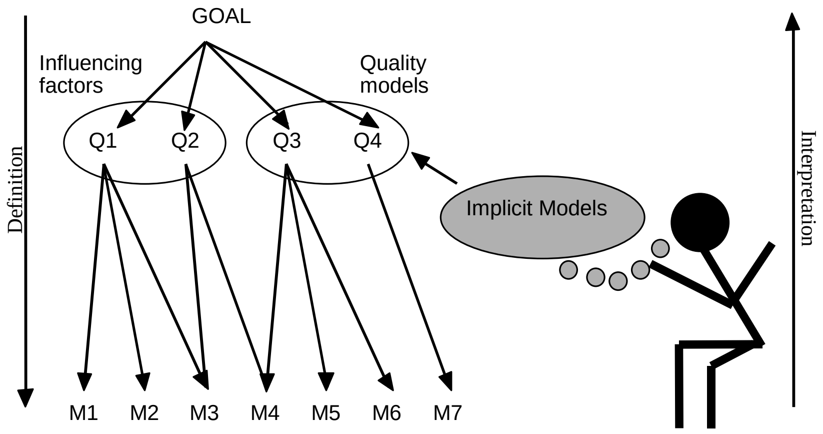


Figure 4. The Goal/Question/Metric approach.

The GQM planning is generally divided into four parts (Briand et al. 1996; Birk et al. 1997; van Solingen & Berghout 1999). Firstly, the measurement goals are defined. Secondly, questions that cover the measurement goals are determined. Thirdly, the measures that need to be collected in order to answer the determined questions are specified. This completes a GQM model. After a GQM model has been specified, it is necessary to develop mechanisms that collect measurement data. This is described in a measurement plan and the associated data collection mechanisms. Tool support is available for the development of the GQM plan, data collection, storage, and visualisation (VTT 1999), as well as data analysis (standard statistics packages like Statistica, SAS, etc.) (PROFES-Consortium 2000). See Paper II on practical experiences on using tool support in a GQM-based measurement program. Paper IV contains a case study of costs and benefits of applying GQM in industry.

2.3 Improvement approaches

It is a common misunderstanding in the contemporary software process community that assessment methods constitute the improvement methodologies. Yet, assessment only provides information of the existing situation to support improvement planning, and sometimes to support improvement monitoring. Having said that it must be conceded that especially those assessment methods that are based on the notion of maturity (or capability) (Humphrey 1987) tend to lead their user into thinking that the higher the maturity, the better, thus suggesting the path for improvement. However, Weinberg (1992, p. 31) claims that "maturity is not the right word for subcultural patterns because it implies superiority when none can be inferred". Weinberg goes on saying, "The quest for unjustified perfection is not mature, but infantile" (ibid, p. 21).

What should be done, then? There are two basic approaches to improvement: analytic and benchmarking (Card 1991). The analytic approach relies on quantitative evidence to determine where improvements are needed and whether an improvement initiative has been successful. Examples of the analytic approach are the Shewhart plan/do/check/act cycle (Shewhart 1939) and its variations (Ishikawa 1985; Zultner 1990; Gilb 1992; Basili & Caldiera 1995). The Shewhart cycle, called also the Deming cycle (Figure 5) shows the procedure for improvement at any stage. It is rooted in the notion of continuous improvement.

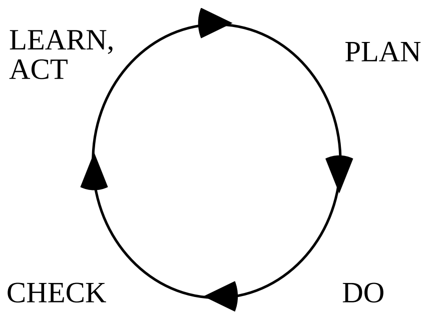


Figure 5. The Deming Cycle (Deming 1986).

PLAN. What changes might be desirable? What data are available? Are new observations needed? If yes, plan a change and test. Decide how to use the observations.
DO. Carry out the change or test decided upon, preferably on a small scale.
CHECK. Observe the effects of the change.
LEARN, ACT. Study the results. What did we learn? What can we predict? (Deming 1986)

Opposite to the analytic approach, the capability models per sé or certification schemes, such as ISO 9000, are related to the idea of benchmarking. Card (1991) states that "the benchmarking approach depends on identifying an "excellent" organisation in a field and documenting its practices and tools. Benchmarking assumes that if a less-proficient organisation adopts the practices and tools of the excellent organisation, it will also become excellent". While this approach may be useful at times, unfortunately the “Get CMM Level 3 by December” attitude is still prevalent in the industry, and resources are wasted for nominal, unnecessary improvements (Bach 1994; van Solingen 2000). Instead, the analytic (or inductive as it is called by Briand et al. (1999)) and the benchmarking approach should be used together. This mix is necessary for process improvement and is acknowledged, e.g. in the quality award criteria such as in the Malcolm Baldrige National Quality Award (MBNQA 2000) where application of both benchmarking and analytic techniques is required. See Paper I for more details and discussion on challenges to quality. However, even with the right mix, successful process improvement is always context-dependent, because "there is no 'right' way to improve quality. Every organisation must come up with an approach that works for them" (Pyzdek 1992).

In summary, instead of using the assessment methods to guide software process improvement effort, selecting the corresponding improvement approach

(McFeeley 1996; ISO/IEC-15504-7 1998) is a good start. The PROFES improvement methodology places a more explicit emphasis on product quality as improvement driver (PROFES-Consortium 2000). PROFES uses a variant of the Quality Improvement Paradigm (QIP) improvement cycle (Basili et al. 1994a), and also integrates assessment and measurement, which are discussed in Papers VI and VII in more detail.

2.4 Discussion

Software process assessment and improvement, and software measurement are all active fields in the software engineering community, both in the academia and in industry. During the past ten to fifteen years whole new disciplines have been established and much has been achieved. On process improvement there are quite a few good, reported experiences (Humphrey et al. 1991; Deutsch 1992; Willis 1998; Ferguson 1999). On the other hand, there are some well-documented limitations and shortcomings of the software process capability based approaches (Bollinger & McGowan 1991; Card 1991; Humphrey 1991; Card 1992; Weinberg 1992), and there are those who prefer heroes over software processes (Bach 1995).

In general, there seems to be a tendency to move from assessment based improvement to measurement based improvement. When organisations start to improve, general guidance that is provided through assessment, is enough. As they continue to improve, organisations need a deeper understanding of their own processes and products. To do this they need measurement data, which helps them find improvement areas. This is also visible in the software process capability models where measurement is in focus at levels 4 and 5. El Emam (1998) even writes that statistical analysis on assessment data shows that the five (or six) levels of capability could be reduced into two – process establishment or measurement as the main focus for improvement.

There are some obvious shortcomings in the current research and practice. Thousands of assessments have been performed (SPICE 1998; SEMA 2000) but little has been discussed on the various kinds of assessments that are needed, i.e. the assessment types, or how the assessments are carried out, i.e. the assessment modes. In addition, the use of measurement data in assessments has been neglected. Measurements have been considered (e.g. Baumert & McWhinney 1992; Park 1996), but so far they have not been fully integrated into the assessment methods or used explicitly to drive the assessment. Further, software measurement seems to be a relatively established field where methods and techniques for measuring software products and processes are commonplace. Managing measurements with organisational and project goals and understanding the nature of software measurement has helped to get better value from software measurement. What seems to be lacking is the use of common frameworks to better understand, manage and utilise measurements. One aim of this research is to show how to use software process reference models for this purpose.

3. Assessment types and modes

3.1 The need for assessment classification

This chapter presents a classification of software process assessment types and modes. The work in this chapter is largely based on the work documented in (Beitz & Järvinen 2000). Assessment methods typically offer officially only one type of assessment (Masters & Bothwell 1995; Dunaway & Masters 1996; Bicego et al. 1998). In practice, several assessment types are needed because organisations use different assessments for different purposes. Further, the assessments are often modified to suit a specific purpose, but the modifications or their rationale are rarely documented or classified. A classification of assessment types and modes is necessary to understand assessments and their role in software process improvement, and especially to position continuous assessment that was of special interest in this research. Continuous assessment is further investigated in Chapter 4.

3.2 Assessment types

Three main types of assessments are defined here: Overview, Focused and Continuous Assessment (Figure 6). The basic idea in the typology is that from Overview to Focused to Continuous Assessment there is an increasing frequency and depth, and decreasing breadth in the assessment. Each of the three main assessment types has their own particular use. Typically, organisations start with an Overview Assessment to get an understanding of their current situation with its strengths and weaknesses. Focused Assessments are then employed to probe the selected processes in more detail. Continuous Assessments can be integrated with existing measurement programs to monitor the selected processes during and after process improvement. There are variants of each of the three main assessment types, which are also discussed.

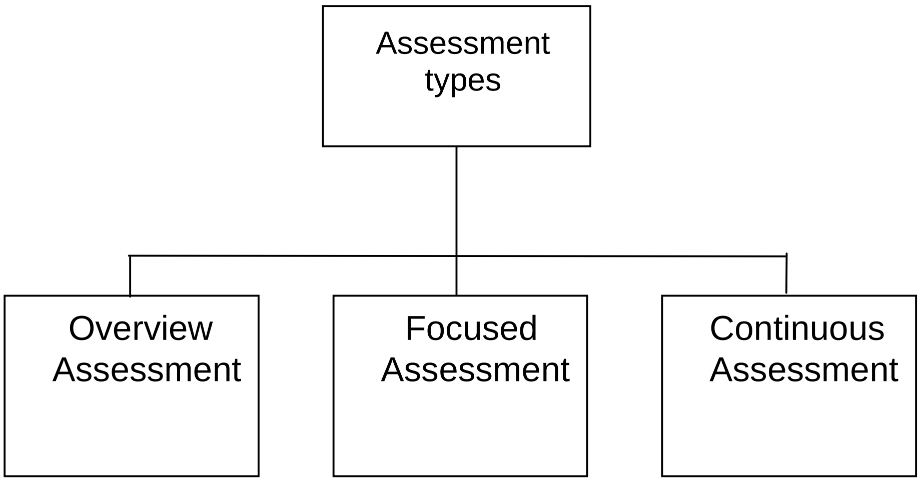


Figure 6. Assessment types.

Each assessment type has it benefits and limitations. In practice, however, the assessment type used will largely depend upon the current state or the organisation, and how it wishes to use assessment to influence its process improvement program. Hence, there should be an understanding before the assessment is started of what must be achieved after the assessment has taken place.

The remainder of this section describes the different assessment types. The format of the descriptions is as follows. Firstly, there is an overall description of an assessment type. Secondly, the advantages and problems are listed. Finally, the variants to the particular assessment type are discussed.

3.2.1 Overview Assessment

Overview Assessment briefly assesses most or all processes at lengthy intervals, e.g. every other year. At minimum, the assessment results may show which processes exist, but will not reveal the capability level of those processes (i.e. how well they are being performed).

Advantages

- Provides a good overview of processes if the organisation has never been assessed or much time has passed since the last assessment
- For organisations with most processes at level 0 or 1 capability this is a low cost assessment to determine the existence of processes
- Fast way to find missing or incomplete processes within the organisation.

Problems

- Does not necessarily measure how well the processes are being performed
- Does not provide a detailed account of process weaknesses.

Full Assessment (Exception)

There is an exception in the typology, viz. Full Assessment. It is included in the typology because it is sometimes done in practice or mistakenly advocated instead of normal Overview Assessment.

In Full Assessment, all processes are examined in great detail. This would mean, for example, assessing all ISO 15504 processes through all capability levels using a detailed set of assessment indicators, e.g. base practices, work products and management practices. As such, Full Assessment is not recommended. While it provides very detailed information from each process, it is usually too expensive and time-consuming to perform. Moreover, it is usually feasible to improve only few issues at a time in an organisation. Therefore, for many of the processes much of the detailed information from Full Assessment will be outdated or obsolete at the time when improvement planning and implementation for those processes will actually be done. Therefore, a less rigorous assessment is normally in order.

3.2.2 Focused Assessment

A Focused Assessment is done to support an improvement program. Typically, it is preceded by an Overview Assessment that provides recommendations for Focused Assessments. These assessments are then synchronised with an overall improvement plan so that a Focused Assessment, a snapshot of maybe just one process, is delivered at a proper time. If a Focused Assessment is done too early there is a risk that the process may change before suggested improvements are implemented, making the recommendations obsolete and causing rework.

Advantages

- Focuses on the most relevant processes in the organisation
- Provides a detailed capability profile to help build and drive the improvement program
- Does not waste time in assessing irrelevant processes that will not impact the organisation.

Problems

- When an organisation has no clear goals, it can be difficult to determine the relevant processes
- May ensue high costs if not focused properly
- Loss of conformance if respective reference models are not covered.

Fixed Assessment

A variant of Focused assessment, Fixed Assessment covers only one or a few processes for prescribed depth, i.e. the depth is decided beforehand, e.g. by a certification body or customer needs. Fixed Assessment is often performed as an audit to ensure conformance. Fixed Assessment is good for organisations that have a fixed set of requirements to be fulfilled, such as the ISO9001 certification or regulatory demands. Examples of typical users of Fixed Assessments are organisations developing mission critical and safety critical systems, where for example IEC 61508 (1998) can be used as criteria for software process intended to produce systems of needed (high) reliability. In addition, a company developing regulated business software, such as handling stock exchange transactions, may also require Fixed Assessment.

3.2.3 Continuous Assessment

Continuous Assessment is a special case of assessment where information from the software development process is used actively to facilitate software process assessment, and help to monitor software process implementation during project execution. In short, the continuous assessment provides a frequently updated structured view of process capability against a reference model. Tentatively, there is no limit to the frequency or breadth of continuous assessment. However, an assessment of all software processes in real-time is currently hardly feasible even if there were some interest for it. Hence, Continuous Assessment is intended to be repeated at selected project intervals, such as product releases or milestones.

Advantages

- Improved visibility of software processes
- Early detection of process deviations
- Reduced cost of assessment
- Once implemented easily manageable.

Problems

- Setup costs.

Measurement based Continuous Assessment (MCA)

Theoretically, Continuous Assessment may be performed without much integration to measurement activities. The idea of Continuous Assessment *per se* is to provide a mechanism to monitor the capability of software process more frequently than that currently offered by software process assessment methodology providers. In practice, however, the link to measurement is vital. Without tight integration to process measurement the feasibility of Continuous Assessment is minimal. Therefore, the subsequent investigation of Continuous Assessment in this thesis is devoted to Measurement based Continuous Assessment (MCA). In brief, a successful implementation of Measurement based Continuous Assessment usually requires:

- focused improvement area
- measurement experience
- adequate data collection infrastructure.

MCA is discussed further in papers V and VII and in chapter 4. See also experiences of MCA in paper VIII.

3.3 Assessment modes

This subchapter describes the different assessment modes, i.e. how the assessment is conducted. The most prevalent assessment mode is Self-Assessment, where an individual, group or an organisation performs an assessment of one's own software processes without much expertise or training on software process assessment. When talking about software process assessment in common parlance, usually Team-led Assessment is intended. In Team-led Assessment the software processes are investigated by a trained internal or external team using a specific assessment method. The main distinction between Self-Assessment and Team-led Assessment is the level of required assessment training and degree of formality in performing the assessment. Emerging assessment modes are Distributed Assessment and Automated Assessment that complement self-assessment and team-led assessment and in certain situations offer advantages over the more traditional assessment approaches.

3.3.1 Self-Assessment

Self-assessment is the most common way of performing a software process assessment (Dutta et al. 1996). There are some methods and tools that are intended for self-assessment purposes (e.g. Steinmann & Stienen 1996; Doiz 1997), which require little or no advance knowledge of software process assessments. The popularity for self-assessment lies in its low cost, good accessibility and ownership of the results. The user does not necessarily have to commit to anything when making a self-assessment using an assessment method or tool available from the public domain. Without commitment and adequate knowledge about assessments, the assessment results are often highly variable (Card 1992; SPICE 1998). For example, as in any technical field the assessment terminology is very specific, and without knowledge of the meaning of assessment-related terms, the interpretations may distort assessment results. Another problem often faced with self-assessment is its limited value for improvement planning, as many of the self-assessment methods do not provide a detailed information on process capability. In addition, without professional advice and insight the self-assessment results may be difficult to interpret.

For results that are more objective self-assessment can be carried out so that the results may be verified. A verifiable self-assessment is promoted by the emerging ISO standard, ISO15504. In practice this means that any competent ISO15504 assessor should be able to verify the assessment results by following the chain of evidence in the assessment records.

Advantages

- Low assessment cost
- Easy access to software process assessment.

Problems

- High variability of results
- Limited value for improvement planning.

3.3.2 Team-led Assessment

Most assessment methods provide support for team-led assessment where an assessment team investigates selected areas of the software process. The assessment is made in a limited time, and it involves personnel interviews and document reviews. The result is a snapshot of the current software process capability that typically is reliable and detailed enough to be used for improvement planning. Some problems with team-led assessment are its potentially high cost depending on the size of the assessment team and interruption of development work due to the requirement for interviews and document reviews. The follow-up of improvement activities may also be difficult if assessment expertise and feedback to personnel is not available after assessment.

For lowering costs, there is a special case of team-led assessment with only one competent assessor under whose supervision the rest of the team is working. However, some methods, such as SEI´s CBA-IPI (Dunaway & Masters 1996) for CMM-SW (Paulk et al. 1993) or Bootstrap Institutes BOOTSTRAP (Bicego et al. 1998), require more than one competent assessor in the assessment team for assuring assessment reliability.

Advantages

- Reliable results
- Good value for improvement planning.

Problems

- High costs
- Obtrusive to development work as personnel are interviewed
- Results only a snapshot of current capability.

3.3.3 Emerging approaches

As software process assessments have become more commonplace in the software industry, more alternative approaches have been sought. For many, especially SMEs (Small and Medium-size Enterprises), the team-led assessments are too expensive or time-consuming. Doing the assessment in a more distributed fashion is expected to bring savings while maintaining an adequate reliability of the results. For some, especially more advanced organisations, the traditional assessment does not provide adequate information for process monitoring purposes or updated status on process improvement activities. Assessment automation may provide more in-depth and real-time information to satisfy these needs.

Distributed Assessment

In a typical assessment information gathering, i.e. interviews and document reviews, takes most of the assessment time. For example in the SPICE trials 47% of the time was spent on gathering evidence (El Emam & Goldenson 1999) and in CMM assessments the number of observations can go up to 3000 (Dunaway et al. 2000, p. 30). In distributed assessment the idea is to spread these human-intensive tasks, e.g. to the people producing the information and among assessors. The role of the assessment team is more that of verifying incoming information than gathering corroborative information. There are instances of improved efficiency by using distributed assessment. For example, a Japanese company was able to perform over 1000 software process assessments a year with an assessment unit of five people (Miyazaki et al. 1995). As another example, the SEIs Interim Profile method uses a distributed questionnaire-based approach for rapid assessment (Whitney et al. 1994) that is intended to be performed between normal CMM assessments.

Advantages

- Assessments may be done efficiently.

Problems

- Ensuring reliability may be problematic.

Automated Assessment

The idea of fully Automated Assessment is that the assessment indicators are integrated in the software process and assessment is done using criteria for interpreting the measurement data. The criteria could be embedded into a tool, such as an expert system. Currently this can work in only very limited and special conditions. For example, if an organisation has a statistically stable process, assessment automation can be considered using the control limits for the process as assessment criteria.

Partly Automated Assessment seems more promising than attempting to fully automate assessment. Using software measurements as supporting evidence for the assessment of software capability opens new possibilities for assessment. Firstly, the interruptions for the personnel and their work can be reduced as more information is acquired automatically online. Secondly, assessments can be carried out more frequently as measurement data is gathered continuously. Thirdly, providing an audit trail for assessment is easier as more judgements are based on measurement data. Finally, assessment cost is reduced, as fewer people are needed to perform an assessment and assessments themselves become an integral part of the job. There are also difficulties associated with partly Automated Assessment. Perhaps the biggest problems are that collecting measurement data is expensive, and building the measurement collection as part of the software process is slow.

Advantages

- Frequent and in-depth assessments are possible
- Interruptions are minimised for personnel and their work
- Experts are still needed to analyse and interpret the results.

Problems

- Extensive indicator metrication is expensive
- Experts are still needed to analyse and interpret the results.

3.4 Summary of assessment types and modes

There is no one “right” way to approach or perform assessment. The different assessment types and modes are complementary approaches that are important as the utility for assessment is broadened in the industry. New capability models (such as Fisher 1998; Niessink & van Vliet 1999; Earthy 2000) are being developed, and integration of software assessment with other domains or more general assessment frameworks continues. For example, Deutsche Telekom is already using results from BOOTSTRAP assessment to replace ISO9001 (ISO/IEC-9001 1994) audits and is proceeding to combine BOOTSTRAP with EFQM (Bergmann 1999; EFQM 1999). In such a dynamic and expanding field it is clear that more is needed than just one kind of of assessment (type), performed the same way (mode) every time.

The use of software measurements for assessment purposes will continue to grow but traditional assessment approaches also remain strong. Not only is it too expensive to metricate all indicators but assessment work also remains intellectual work that can only be made by and with human experts. Further, there are signs, such as interest to techniques like PSP and TSP (Humphrey 1997; Humphrey 2000), that software measurement is slowly becoming an integral part of software engineering work. The benefits of this are already visible in several state-of-the-art companies (Curtis 2000; Paulk et al. 2000). When people are skilled and empowered to define and analyse their own metrics, continuous assessment can yield extra benefits for understanding, controlling and improving not only the software processes but also this knowledge intensive work in general.

4. Measurement based continuous assessment

4.1 Principles of measurement based continuous assessment

Software process assessment is carried out to determine the status of the software processes comparing them with a reference model such as ISO 15504 or CMM. The current prevalent practice is that the assessment team carries out an overview assessment (Figure 7) at infrequent intervals, perhaps every other year. This assessment requires significant effort, including multiple interviews and document reviews. The assessment leads to recommendations for improvement that are subsequently prioritised and implemented over time.

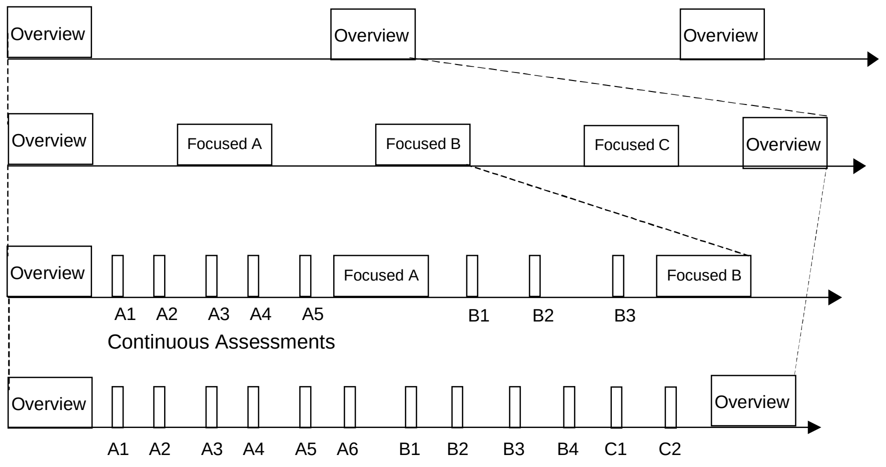


Figure 7. Assessment scenarios.

A focused assessment may then be performed on those processes selected for improvement. Such focused assessments naturally require fewer resources to perform, but are still conducted in a traditional manner. Continuous assessment, on the other hand, employs a different paradigm for conducting assessment.

The idea of continuous assessment is to collect information from the software process as it becomes available during software engineering work and make continuous assessments (Figure 7) at selected intervals, such as project milestones. The continuous assessments can provide information for focused assessments and sometimes even replace them. It is still a good idea to make overview assessments, every other year for example, and use them to have a general impression of all processes.

4.1.1 Assessment as a measurement instrument

There are various ways to implement continuous assessment, for example in a process-centred development environment or through intensive data collection procedures. The approach in this research is to use continuous assessment as a measurement instrument that complies with the GQM paradigm (Basili et al. 1994b), i.e. by conducting continuous assessments using goal-oriented measurement data. An illustration of the information flow between assessment and measurement program is presented in Figure 8. The white areas in the GQM bar represent GQM planning, and the grey areas represent execution of the measurement program. The solid arrows signify flows of information for measurement planning purposes, and the dotted arrows represent the flow of measurement data for capability assessment purposes.

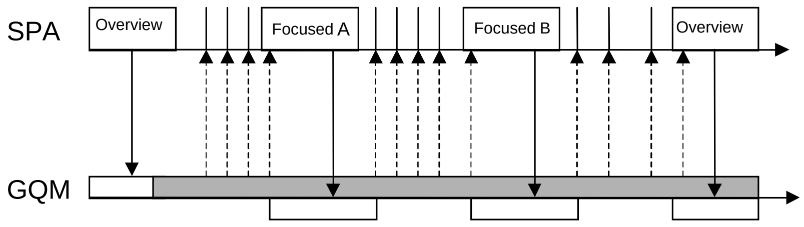


Figure 8. Information flow between assessment (SPA) and measurement program (GQM).

The process assessment results are seen as a set of metrics for the measurement program. Software process assessment is conducted using a specific process reference model and rules for assessment, and for calculating results. Therefore, it can be argued that assessment results are measurements, even if they are complex measurements. They can then be used in a goal-oriented measurement program as any other measurements – to answer specific questions. In practice, this means adding a goal or subgoal to the GQM plan, for example to analyse the system test process by understanding the factors affecting process capability.

4.1.2 Using a reference framework for MCA

A prerequisite for continuous assessment is that a mapping exists between actual measurements and a reference model for software processes. In this research the forthcoming ISO 15504 standard on software process assessment has been chosen as the framework for software best practice, and as the reference model for software process capability. The use of other reference models, such as CMM, is also possible but is beyond the scope of this thesis.

When the ISO 15504 reference model is enhanced with the assessment model defined in Part 5 of the standard, it is possible to find links between actual measurements and the ISO 15504 framework (See Figure 3 on page 28).

Specifically, the assessment indicators provide adequate details for connecting process information to the framework. Process performance indicators are used to determine whether a process exists in reality. For example, the software design process (ENG.1.3 in ISO 15504 reference model) is considered to exist if it can be determined that documents exist specifying:

- Architectural design that describes the major software components that will implement the software requirements
- Internal and external interfaces of each software component
- Detailed design that describes software units that can be built and tested
- Consistency between software requirements and software designs.

If a software design process is functioning in an organisation, it should be straightforward to determine the existence of documents that satisfy the goals listed above. For example, this information can be found in a document management system that tracks the documents produced with a specified process. A report from this system can then help an assessor to determine whether the software design process is being performed.

After determining the existence of a process, the ISO 15504 indicators can then be used to determine the capability of an existing process. Linking information from the actual measurements to management practices, practice performance characteristics, resources, and infrastructure can help an assessor to determine how well the process is performed in relation to ISO 15504. For example, the performance management attribute 2.1 of ISO 15504 Level 2 can be considered as fulfilled if:

- Objectives for the performance of the process will be identified, for example, schedule, cycle time, and resource usage
- Responsibility and authority for developing the process work products will be assigned
- Process performance will be managed to produce work products that meet the defined objectives.

Generally, it is more complex to use actual measurements to assess process capability than using them to demonstrate that processes exist.

4.1.3 Adaptation and reuse of process capability metrics

The ISO 15504 reference framework and assessment model with its assessment indicator set can be utilised to map and reuse actual measurements related to process capability. For optimal results it is recommended, however, that organisations tailor their own indicator sets that map to the ISO 15504 reference model. The indicator set defined in ISO 15504-5 is a generic set that is intended to be used as guidance and a starting point. The adaptation effort does not need to be extensive, but at least the suitability of available indicators should be ensured.

The adaptation starts by mapping the ISO 15504 indicators to the relevant items in the organisation, for example differentiating between embedded systems development and office software. With a customised process capability indicator set, an organisation can focus more specifically on the problems in their processes, and continue to refine the indicators for better precision and coverage.

4.1.4 Granularity of process capability metrics

A reference framework, such as ISO 15504, is often a hierarchical construct aimed at managing complexity. At the lowest levels of hierarchy, the number of data elements can be very high. For example, for the exemplary assessment model contained in ISO 15504 there are hundreds of references to very specific properties and characteristics of work products and management practices. Are all of them equally important? Probably not. An exploratory study done at Tokheim suggests that choosing specific key indicators can provide adequate information on process conformance and capability (PROFES-Internal 1997 - 1999). For operational purposes it seemed to be enough when the life cycle of developing a software feature is tracked with checklist items related to the major events of the feature development. For example, if a functional specification has been approved, it can be assumed that the functional specification has been written and it has been reviewed. Obviously, more thorough checks are needed to ensure that the process works in detail as intended as the occurrence of high level approval does not per se ensure adequate fulfilment of related lower level tasks. It is beyond the scope of this thesis to investigate this further, but these key indicators or "super metrics" show a promising direction for increasing the feasibility and applicability of MCA in the future.

Another issue to be remembered with software measurement is the capability of the organisation and its processes (Kitchenham 1996, p. 109). According to Zubrow (1997), there are certain types of measurements that suit best for a given capability. For example, it is not a good idea to try to compare productivity in Level 1 projects as the processes are not likely to produce consistent data that could be used for decision making.

The trends in measurement evolution that Zubrow (1997) lists include

- Project → Product → Process
- Post-release → In-process
- Prediction → Status
  → Control
- Univariate → Multivariate
- Implicit models → Explicit models
- Descriptive → Inferential.

These and other issues related to measurement and capability are important when planning for measurement evolution but are beyond the scope of this thesis. In conclusion, the type, quantity and granularity of process capability metrics should be suited for the assumed capability level in order to be most effective.

4.2 A method for measurement based continuous assessment

This section describes a method for measurement based continuous assessment. The method has been motivated and constrained by the requirements of the industrial application cases in the PROFES project (PROFES-Internal 1997 - 1999), which aimed to ensure the practical applicability of continuous assessment. The multi-layer review process in the PROFES project (Järvinen et al. 2000) was used to assure the quality and usability of the method.

Steps for applying Measurement based Continuous Assessment

There are six steps for applying measurement based continuous assessment[1]. Its prerequisites are that at least one overall assessment has been made previously, and that goal-oriented measurement is being planned or revised. It is difficult to select a limited set of processes if the overall capability is not known. In practice, experience has shown that continuous assessment is likely to have the highest cost/benefit ratio when used to augment an existing goal-oriented measurement program (Paper VIII. See also van Veldhoven 1999).

The six steps to apply continuous assessment are as follows:

I Select processes to be examined

II Construct or update measurement goals

III Define indicators for process existence and capability

IV Construct or update measurement plans

[1] The words "measurement based continuous assessment" and "continuous assessment" are used interchangeably in this section.

V Collect data and assess selected processes

VI Analyse results and do corrective actions.

After step VI, it is possible to continue starting again from any of the steps depending on the given situation.

I Select processes to be examined

The principle in selecting processes for continuous assessment is that only those processes that are either critical or currently being improved are included. Generally, it is worth starting with just one or two processes in order to gain experience of continuous assessment. In short, prospective processes for continuous assessment are usually those that a) have already been measured, b) are being, or planned to be improved, and c) are supported by tools to minimise manual data collection. The selected processes should then be prepared for continuous assessment so that:

- A target rating is recorded for each practice, which can be the same as the current rating if only monitoring is attempted. This is the starting point for systematically governing improvement activities.
- Applicable sources for measurement data are defined. Examples of good data sources with the potential for automatic data collection are Lotus Notes, MS-Project, any configuration management system, or any database that is used to collect project data, e.g. defect database (Parviainen et al. 1996). However, the data does not always have to be automatically collectable, although this is usually preferred.

II Construct or update measurement goals

The measurements relating to process existence and capability are typically integrated into an existing measurement program. Therefore, the measurement goals are updated, or new measurement goals need to be constructed to accommodate the capability-related metrics.

III Define indicators for process existence and capability

For each selected process, the most important measurements are those indicating whether the process is performing or not, is producing useful results and is fulfilling its purpose. This is the ISO15504 process dimension. Depending on the scope chosen, the metrics related to the ISO15504 capability dimension can also be reviewed. These metrics are used to measure control, management, and improvement aspects of the process. However, there are practices that are better left for assessment interviews, as it is usually not appropriate or feasible to cover everything automatically. For example, it is easier to ask a person about his or her job satisfaction than to construct automatic measurements to inquire that.

III a) Define process existence indicators

The ISO15504 process dimension includes base practices that are the minimum set of practices necessary to successfully perform a process. For example, the base practices for the Software Construction Process (ENG.1.4) that covers coding and unit testing in a software life cycle are: Develop software units, Develop unit verification procedures, Verify the software units, and Establish traceability (ISO/IEC-15504-5 1998). Metrics suitable for base practices are usually those that give evidence of the base practice existence, i.e. that enough work that contributes to fulfilling the purpose of the process has been done. Information is usually found in the artefacts, which are the work products that are produced in the process although this is not strictly required by ISO 15504 at Level 1.

III b) Define process capability indicators

The ISO15504 capability dimension should also be examined for the selected processes. The ISO15504 capability dimension contains information on how well the practices are performed, and how well the process runs. Usually, going through Level 2 of the capability dimension is enough, as this is the present state of the practice. Recent SPICE assessment trials results show that only 12% of process instances (341 in total) were higher than Level 2 (SPICE 1998). Naturally, higher levels can be revisited depending on target capability. Information for identifying the capability dimension can mostly be found in the project plan, project reporting documents, configuration management system, and the actual work products.

IV Construct or update measurement plans

The definition of relevant measurements for continuous assessment does not necessarily require using a goal-oriented measurement plan with goals, questions, and associated metrics, as the ISO15504 processes form the structure for the investigation. However, an existing GQM plan is an excellent source of information. Some of the GQM measurements may also be used to facilitate software process assessment. Augmenting an existing measurement program with process capability focus provides added value at reasonable cost. For example, it is possible to monitor process improvement activities closely and evaluate the effectiveness of the process changes.

The integration of measurement activities into the software process must be planned with care. Usually this involves at least minor changes to the process, as data must be recorded or structured in a way that is processable later. Software tools and databases are a key source for process data, but even then some effort is needed to structure, convert, and extract data from various tools and databases. Some data may also be entered manually from questionnaires or checklists. Within the PROFES project, various checklists proved to be particularly useful for the continuous assessment trials. See the Tokheim example in Chapter 5 for more information on the use of checklists for continuous assessment.

V Collect data and assess selected processes

The data for continuous assessment indicators should be collected during project implementation as part of the data collection routines agreed in the measurement plan. A spreadsheet program such as Microsoft Excel may be sufficient for data consolidation and analysis, but more sophisticated tools such as MetriFlame (VTT 1999) are often useful. The frequency of continuous assessments varies, but project milestones and GQM feedback sessions are typically good candidates for timing a snapshot of process capability. Please note that for some indicators there may be measurement data available, but for others a quick check on the process by a competent assessor is needed, as it is not cost-efficient to automate everything.

VI Analyse results and do corrective actions

The assessment results from the continuous assessments are discussed and interpreted in GQM feedback sessions, similar to any other measurement results prepared for the feedback sessions. After analysing the data, specified corrective actions are taken and data collection is continued. The measurement program needs to be analysed critically and altered or even discontinued whenever appropriate.

4.3 Techniques for MCA

When the idea for facilitating process assessment with measurement data came up it was assumed that high process capability and extensive tool support are needed for applying MCA. However, practical exploration of the possible means for achieving MCA showed that there are simple techniques that can be used even in modest environments (Paper V).

For the interest of feasibility and cost, it is recommended that the measurement program for continuous assessment contains three elements for data collection:

- direct measurements,
- event driven measurements, and
- work product driven measurements.

Conceptually, these elements correspond to the exemplary assessment model of the ISO 15504 framework for process assessment (ISO/IEC-15504-5 1998). More specifically, the measurements relate to the work products and management practices associated with the processes. See examples of these measurements in Chapter 5.

Direct measurements can be obtained more or less directly from the software tool environment. Event driven and work product driven measurements are mostly checklists that are filled by the relevant people when the stage in the process is right. As there is always some effort associated with filling checklists – be they in manual or automated format – the measurement collection should always support or at least follow the work at hand. Optimally, the expertise and judgement of people is exploited so that their data collection effort is not trivial but adds value and depth to the measurement. For example, asking opinions on process and product quality or judgement over complex process relationships can provide additional insight for understanding the software process even if the answer is just a tick in a checklist.

Finally, it should be acknowledged that in most cases it is not feasible or cost-efficient to try to collect all relevant information through a measurement program. There are many aspects of the software process where an interview performs best, for example to determine whether “commitments are understood and accepted, funded and achievable” (ISO 15504-5 1998, p. 34).

4.4 Tool support for MCA

Automation of measurement data collection and management enables measurement data to be used more cost-efficiently. Proper tool support is essential when aiming to reduce the work necessary for the measurement tasks. Three tool categories to support the MCA approach are introduced:

- Support for managing measurement plans and data
- Support for mapping measurement data to reference framework
- Support for monitoring process capability.

Firstly, support for managing measurement plans and data is needed to establish and run a measurement program. Due to the focus of this thesis, the support for GQM is emphasised. Secondly, as measurement data becomes available, it needs to be mapped to the chosen reference framework – ISO 15504 in the case of this thesis. Thirdly, as MCA is done frequently, support for monitoring process capability is needed. Note that there are numerous other issues related to supporting software measurement, which fall outside the scope of this thesis. See (Grady & Caswell 1987; Fenton & Pfleeger 1996; Kitchenham 1996; Florac & Carleton 1999) for more information.

4.4.1 Managing measurement plans and data

Support for managing measurement plans and data is essential for long-term success of a measurement program (Fenton & Pfleeger 1996). Van Solingen and Berghout (1999, p. 71) describe that the activities needed from a measurement support system include: collecting, storing, maintaining, processing, presenting and packaging measurement data. Kempkens et al. (2000) discuss measurement program support further and present a framework for setting up a tool support.

There are several tools that support parts of a measurement support system such as spreadsheets, statistical tools, database applications and presentation tools (van Solingen & Berghout 1999, p. 70). The integration of these tools with existing data sources is important (Kitchenham 1996, p. 109). There are some tool environments, such as SAS Data Warehouse, which can be tailored for this purpose; however, MetriFlame is so far the only one that is built especially to support the GQM (van Solingen & Berghout 1999, p. 70).

MetriFlame is a tool environment for managing measurement plans, data, and results. MetriFlame is suitable for measurement data collection, metrics definition and calculation, and the presentation of analysis results in various formats. Documents and databases created during a normal software development process are typical sources of measurement data. (VTT 1999)

The main elements of the MetriFlame tool environment are (Figure 9):

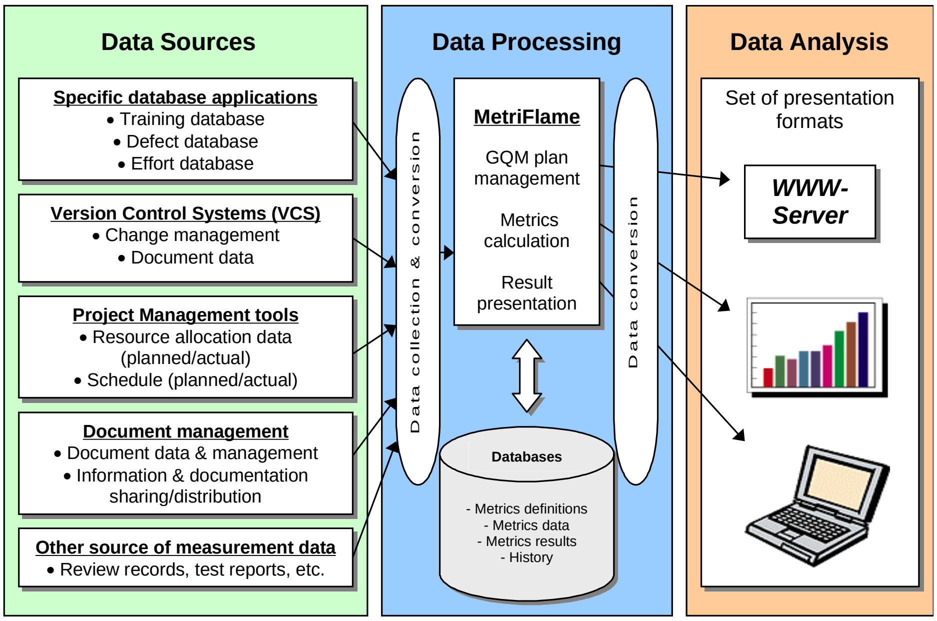


Figure 9. MetriFlame tool environment.

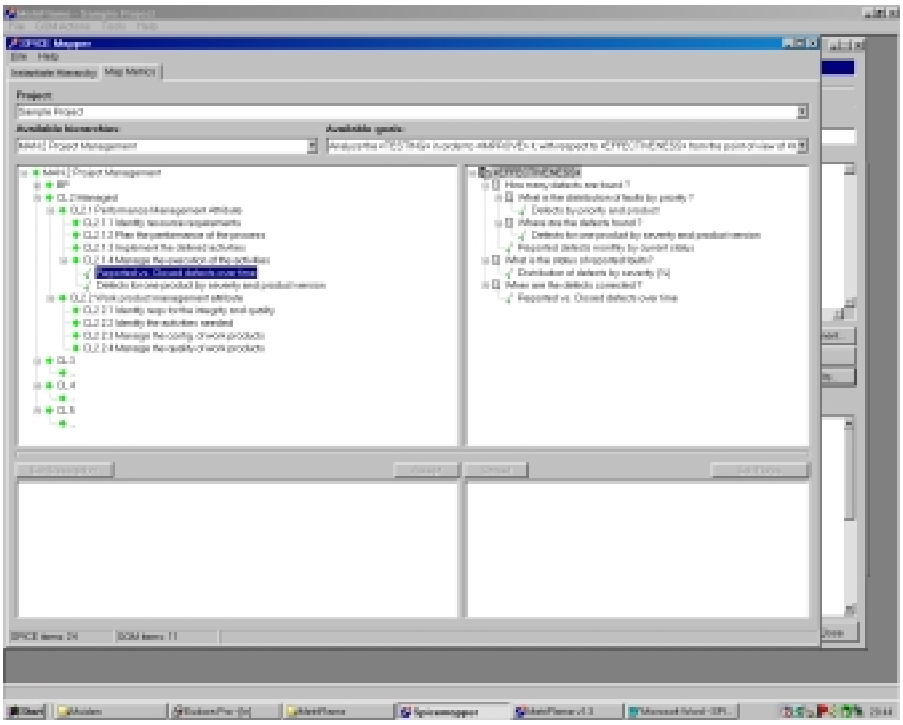
- Measurement data collection and conversion components (data sources).
- The MetriFlame tool (data processing)
- Display formats for metrics' results (data analysis).

It is possible to automate measurement data collection and analysis with MetriFlame, and to support measurement programs where metrics may vary from project to project. MetriFlame metrics calculation is based on the evaluation of associated formulas. Once the formulae are filled out with values and the latest data is available, the measurements can be repeated. This reduces the need for extra work each time the measurement results are calculated.

Noteworthy for MCA, the data sources presented in Figure 9 represent the information source types needed to cover the information content ISO 15504 reference framework (Parviainen et al. 1996).

4.4.2 Support for mapping measurement data to reference framework

The support for mapping measurement data to reference framework is important for MCA. This enables on-line monitoring of selected software processes using related measurement data. This is especially useful for the purposes of MCA where measurement data plays an important role in the frequent assessments. SPICE Mapper (VTT 2000) is an extension to the MetriFlame tool environment. With SPICE Mapper it is possible to link measurements in a GQM plan to the ISO 15504 processes and base practices (Figure 10). In this research ISO 15504 has been the selected reference model for software processes but other reference models, such as CMM (Paulk et al. 1993), ISO 12207 (1995) or IEC 61508 (1998) can also be used. The reference model just needs to be constructed with the hierarchy tool included in the SPICE Mapper.



SPICE Mapper

Instantiate Hierarchy | Map Metrics

Project: Sample Project

Available hierarchies: MAN.2 Project Management

Available goals: Analyse the <TESTING> in order to <IMPROVE> it, with respect to <EFFECTIVENESS> from the point of view of

```
MAN.2 Project Management
  BP
  CL 2 Managed
    CL 2.1 Performance Management attribute
      CL 2.1.1 Identify resource requirements
      CL 2.1.2 Plan the performance of the process
      CL 2.1.3 Implement the defined activities
      CL 2.1.4 Manage the execution of the activities
        Reported vs. Closed defects over time
        Defects for one product by severity and product version
    CL 2.2 Work product management attribute
      CL 2.2.1 Identify reqs for the integrity and quality
      CL 2.2.2 Identify the activities needed
      CL 2.2.3 Manage the config. of work products
      CL 2.2.4 Manage the quality of work products
  CL 3
  CL 4
  CL 5
```

```
EFFECTIVENESS
  How many defects are found ?
    What is the distribution of faults by priority ?
      Defects by priority and product
    Where are the defects found ?
      Defects for one product by severity and product version
    Reported defects monthly by current status
  What is the status of reported faults?
    Distribution of defects by severity (%)
  When are the defects corrected ?
    Reported vs. Closed defects over time
```

SPICE items: 24 | GQM items: 11

Figure 10. *Mapping GQM metrics to ISO15504 practices with SPICE Mapper.*

4.4.3 Support for monitoring process capability

Monitoring of process capability over time is interesting for organisations committed to long term process improvement. Process capability trend analysis will make the effect of process changes on process capability visible over a given period. Process capability trend analysis becomes especially interesting with MCA as successive assessments are done more frequently than with traditional approaches. Thus, the trend lines are more up-to-date and closer to daily work. The PROFES Capability Trend Analysis Tool (Etnoteam 1998) has been created in the PROFES project to support monitoring of process capability trends.

Figure 11 shows how process capability over time can be examined with the tool. This figure shows especially how it is possible to drill-down into individual process attributes for the trend analysis. For MCA, the process attribute level is often interesting to examine as rapid assessment cycles can be targeted towards monitoring of very focused improvement activities. This added level of detail has been missing from the assessment tools currently available.

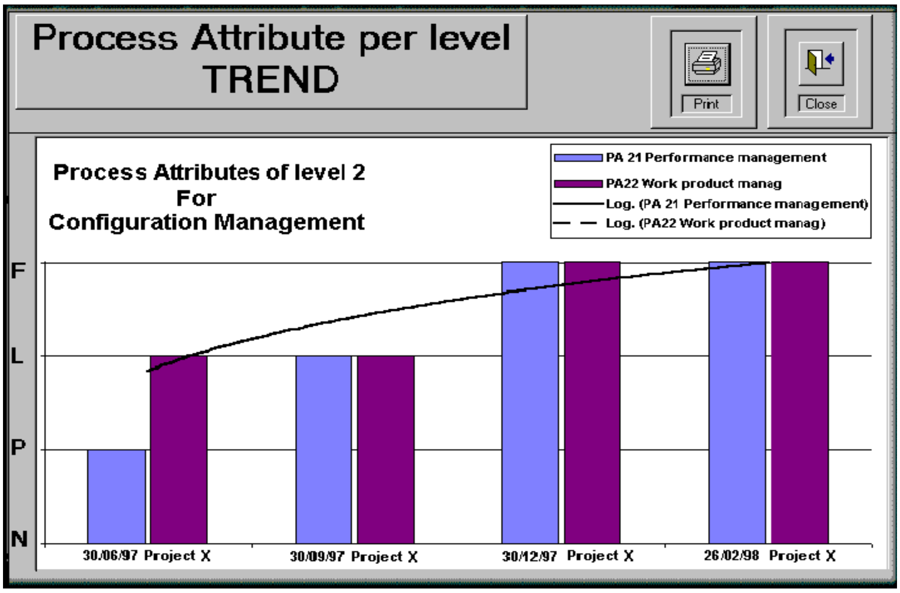


Figure 11. PROFES capability trend analysis tool.

4.5 MCA vs. related work

4.5.1 MCA vs. CMM

There are some elements in the CMM that point towards using measurement results for assessment purposes. There is the Interim Profile that can be produced between regular CMM assessments (Whitney et al. 1994). The Interim Profile is based on using questionnaires for information and gives a snapshot of the selected KPA capability. There is a mapping between the CMM practices and the potential indicators and measurements (Park 1996). This measurement map provides much information for doing assessments of selected practices aided by measurement data.

4.5.2 MCA vs. ISO 15504

In general, ISO 15504 emphasises the traditional team-based approach to assessment. There are some hints in ISO 15504 toward using the assessment indicators with a more measurement based approach but it is left for the user to define how this could be done. However, the structure of the exemplar model in Part 5 is well suited for measurement adaptation. In parts 4 and 5 of an earlier version of the ISO 15504 (SPICE-4 1995; SPICE-5 1995), the guidance of using continuous assessment as an alternative assessment paradigm was more significant and could be considered useful, but was removed from the final versions of ISO 15504 (1998).

4.5.3 MCA vs. BOOTSTRAP

The Bootstrap Institute does not officially recognise the possibility of using BOOTSTRAP in a measurement based fashion. However, the clear structure and detailed assessment indicators of BOOTSTRAP provided a good opportunity to use BOOTSTRAP in the industrial trials of the MCA approach. Special mappings were created to provide a clear projection between measurements and the base and management practices of the selected BOOTSTRAP processes. See Paper VIII on details of an industrial case study using BOOTSTRAP with the MCA approach.

4.5.4 MCA vs. GQM

The GQM paradigm interfaces with MCA well by offering a good solution for handling assessment related measurements in both conceptual and practical sense. Conceptually, GQM provides the framework for the hierarchical assessment data that is structured by the chosen assessment reference model, ISO 15504 in this research. In addition, as in the direct software measurement, assessment deals with people and their work, and GQM helps to address these important issues. Practically, it is advantageous that there are tried and tested techniques in GQM implementation that can also be used for MCA purposes. See Papers III, V and VIII for detailed description on using GQM for assessment purposes.

4.5.5 MCA vs. Improvement approaches

MCA can probably be integrated with the most improvement approaches, as they normally do not address the level of implementation MCA deals with. A closer relationship to MCA can be found from improvement approaches that use GQM. These include the Quality Improvement Paradigm (QIP) (Basili et al. 1994a), CMM–based ami (Pulford et al. 1996), the more generic Pr^{2}imer (Karjalainen et al. 1996) and the product focused RPM (van Solingen 2000). All of these approaches use GQM as their chosen approach for measurement. Finally, the PROFES improvement methodology (PROFES-Consortium 2000) connects in an even closer way to MCA as it also integrates assessment and measurement, which are discussed in Papers VI and VII in more detail.

5. Applying continuous assessment in an industrial setting

5.1 Case background

Tokheim is a worldwide leader in providing systems and services for self-service fuel stations. Tokheim had a revenue of 750 million US$, and 4,800 employees in 1999.

The Tokheim software development centre in Bladel runs the OMEGA project that aims at the functional extension of an existing fuel station management system. OMEGA is a retail automation system designed and developed specifically for the needs of fuel station managers and operators. OMEGA is both modular and configurable from a simple fuel pump console to a comprehensive multi-Point-Of-Sales (POS) configuration with dedicated 'Back Office' workstation for site management purposes. OMEGA is a networked PC-based embedded system. Proprietary hardware is included to perform communication with the fuel dispenser calculators, outdoor payment terminals, vehicle identification hardware and other external equipment on the station forecourt. The main part of the system functionality is developed in software.

The involvement of the OMEGA development project in this case was limited to the OMEGA system test group. Subsequently, the system testing process was chosen for examination (I Select processes to be examined[2]). This independent group performs integration and system test of OMEGA before it is released to the customers. The test group executes general regression tests and focused feature tests for the system. The OMEGA test group also tests country specific properties of the OMEGA system such as currency, language and governmental requirements. The OMEGA system test group had previous experience and expertise in GQM–based software measurement.

[2] The Roman numeral references in brackets in this chapter refer to the MCA steps presented in Chapter 4.2 starting from page 52.

5.2 Finding indicators for continuous assessment

To establish a setting in which continuous assessments can be performed, the existing measurement program and the existing process improvement program had to be integrated. Hence, the system testing process as defined by BOOTSTRAP was investigated to find relevant goals, questions, and metrics for the process (II Construct or update measurement goals). It was assumed that GQM plans based on the ISO15504 reference model are generic, and therefore need to be created only once. These generic GQM plans would then be available in a future implementation of continuous assessment. After this, the assessment indicators for the system testing process were adapted to suit Tokheim, and specifically the OMEGA environment (III Define indicators for process existence and capability). This customisation process is illustrated in Figure 12 and resulted in:

- A direct measurement data collection plan – a set of metrics that were related to the ISO15504/ BOOTSTRAP assessment indicators, and could be collected directly using software development tools
- A set of document checklist items that needed to be checked for each development document, as they were related to the ISO15504/ BOOTSTRAP work product indicators
- A set of event-driven checklist items that needed to be checked for a specific event that occurred (i.e. system test is started, or a defect is found), and were related to the ISO15504/ BOOTSTRAP assessment indicators.

This customisation of assessment indicators was expected to be project-specific. However, a comparison with another Tokheim project indicated that the altered indicator set of OMEGA could be largely reused for other projects, although they may be conducted in a different application domain.

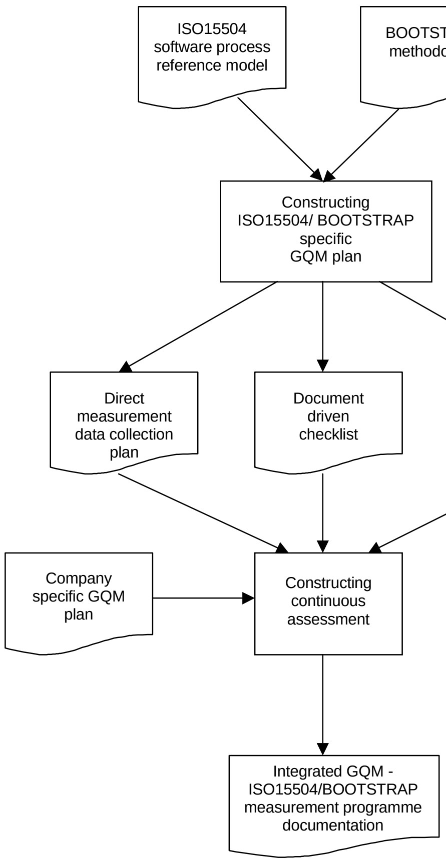

ISO15504 software process reference model
BOOTSTRAP methodology
Constructing ISO15504/ BOOTSTRAP specific GQM plan
Direct measurement data collection plan
Document driven checklist
Test driven checklist
Company specific GQM plan
Constructing continuous assessment
Integrated GQM - ISO15504/BOOTSTRAP measurement programme documentation

Figure 12. Integrating GQM measurement and ISO15504/ BOOTSTRAP assessments at TOKHEIM OMEGA.

Another starting point for applying continuous assessment was the existing measurement program. Hence, integrating assessment indicators with measurement programs was done by studying the GQM plan on OMEGA system testing and deciding what aspects of process capability could provide additional information to answer the questions related to the selected measurement goals (IV Construct or update measurement plans). The result was an updated GQM plan (Figure 13) in which the software process capability measurements were integrated.

GOAL

Analyse the	System Testing process
For the purpose of	Understanding
With respect to	- Balancing Cost, Time and Quality
	- Process Capability
From the viewpoint of the	Test group and Test group manager
In the context of the	TOKHEIM OMEGA projects

QUESTIONS

Q-1. What are the preconditions for good testing?
Q-2. What are the costs of testing?
Q-3. What is the impact of the testing process on product quality?
Q-4. What is the duration of the test process?
Q-5. What are good criteria for the decision to stop testing ?
Q-6. What are the rules of thumb on the test process?
Q-7. Are aggregates of system units built?
Q-8. Are tests for system aggregates developed?
Q-9. Are system aggregates tested?
Q-10. Are tests for the system developed?
Q-11. Is the integrated system tested?
Q-12. Is the customer documentation updated?
Q-13. Are joint reviews held?
Q-14. Is the performance planned?
Q-15. Are defined activities implemented?
Q-16. Is the execution managed?
Q-17. Is the quality of work products managed?

METRICS

M.1. Availability of documentation
M.2. Perceptive quality of each type of documentation
M.3. Effort spent per feature on testscript selection and definition
M.4. Effort spent per feature on testscript execution
M.5. Effort spent per feature on failure solution
M.6. Effort spent per domain on testscript selection and definition
M.7. Effort spent per domain on testscript execution
M.8. Detection rate (number of failures detected per hour per person)
M.9. Number of Fatal, Major, Minor & Cosmetic failures
M.10. QPR Status

Figure 13. Continues on next page.

M.11. Number of failures detected during field tests:
M.12, M13. Number of failures per domain and per feature
M.14. Reason why not found during test
M.15. Total duration of testing cycle for full-release
M.16. Duration of testing cycle for pre-release
M.17. Duration between failure detection and solution
M.18. Number of features per pre-releases
M.19, M.20. Duration of feature test and domain test
M.21. Number of features tested per hour
M.22. Total effort spent on testscript selection and definition
M.23. Total effort spent on testscript execution
M.24. Total time spent on testscript selection and definition
M.25. Total time spent on testscript execution
M.25. Total duration testscript execution
M.26. Percentage of reusable features
M.27. Number of test scripts used (names)
M.28. Number of Kilo Lines Of Code (KLOC) of integrated system
M.29. Number of risks identified
M.30. Number of test scripts executed
M.31. Number of tests in schedule
M.32. Number of changes made in the planning
M.33. Number of reviews executed

Figure 13. Tokheim OMEGA GQM plan with metrics for continuous assessment.

Some of these metrics were already used in the original measurement program, but also provided relevant information for the assessment. For example, the measurement program measured the frequency and effort spent on regression tests and this data was collected by the testing report. These measurements could also be used as output work product indicators of the testing process.

5.3 Using measurement data for continuous assessment

The approach for gathering measurement data for the measurement program of OMEGA was that of using multiple ways of data collection as illustrated in Figure 12. Some data were collected directly from the development tools. For example, the data for the failure severity was retrieved from the QPR (Quality Problem Report) database. Some data came from interviews, but mostly the data was collected using checklists embedded into the development process. An example checklist for a document is shown in Figure 14.

Integration / System test script	✔/✘
• Is the test script template used and filled in correctly and completely?	
Integration test strategy / plan	✔/✘
• Is the purpose of integration defined?	
• Does a validation of a subset of the system exist?	
• Does a validation of the integration of the software to other SW components exist?	
System test strategy / plan	✔/✘
• Does it identify a strategy for verifying the integration of system components as defined in the architectural specification?	
• Does it provide test coverage for all components of the system?	
1. Software	
2. Hardware	
3. External interfaces	
4. Installation activities	
5. Initialisation	
6. Conversion programs	
Release strategy / plan	✔/✘
• Does it identify the functionality to be included in each release?	
• Does it map the customer requests, requirements satisfied with particular releases of the product?	

Figure 14. A document checklist for a test script.

The information gained with these multiple means was viewed through the GQM tree structure and the integrated ISO15504 reference model (V Collect data and assess selected processes). A competent BOOTSTRAP assessor verified and examined the collected data, and carried out some clarifying interviews to ensure that the impression of the measurement data was correct. Then he rated the process practices, recorded findings, and generated a process rating profile. This process profile was discussed and analysed in a GQM feedback session along with other material from the measurement program (VI Analyse results and do corrective actions).

5.4 Experiences

A significant finding was that 50% of the company-specific metrics for continuous assessment were in the direct measurement data collection plan (Figure 12). This means that many metrics could be constructed using measurement data directly from the system test process. It was equally noteworthy that the GQM measurement program already running in the company covered 85% of the ISO15504 base practices. Thus, the ISO15504 framework of software processes and assessment indicators served not only as a checklist for the OMEGA system test measurement plan, but also provided useful additions.

Another important finding from the continuous assessment application at Tokheim is that there is a clear need for a description of the ISO15504 assessment indicators in GQM format. Such a description was not available and needed to be developed. Creating the goal, questions, and metrics for one software process consumed 60 person hours of effort (Table 3). This work has to be done once per process and the result can be applied to any other continuous assessment. The actual construction of an integrated GQM plan for continuous assessment took 30 hours. Note that the OMEGA project started to carry out continuous assessments with a good background in measurement, but with limited assessment experience. In practice, a Tokheim employee established the continuous assessment without much prior exposure to software process assessment or measurement. However, measurement and assessment experts at Tokheim guided his work. Hence, the effort required using the ISO15504-specific GQM plan – applying and customising it to the local needs of an organisation and project team seems to be quite reasonable.

Table 3. Example of the effort spent for establishing continuous assessment when starting from scratch.

Activity	**Total effort**	**Effort**
BOOTSTRAP specific GQM plan:	~100 hours	
Learning BOOTSTRAP:		20 hours
Learning GQM:		20 hours
Defining goals and questions:		10 hours
Defining metrics:		20 hours
Defining checklists:		30 hours
Constructing Continuous Assessment:	~30 hours	
Investigate GQM plan:		5 hours
Comparing plans:		15 hours
Integration:		10 hours

The additional cost for collecting and analysing continuous assessment data in the measurement program was approximately 5%. This is a small figure as the cost of a measurement program for the project team is typically 1 - 2% of their total working time (van Solingen & Berghout 1999, p. 33, 35)[3]. However, with an unclear focus or insufficient infrastructure for data collection it is likely that MCA would cause significant overhead. On the other hand, it was found that a sufficient infrastructure for data collection does not necessarily imply state-of-the-art tooling or large overhead in manual data collection as using checklists

[3] See paper IV for more discussion on applying GQM in industry.

(Figure 12 and Figure 14) embedded to the process was effective and efficient for measurement data collection.

The continuous assessment approach provided added value for Tokheim. For example, mapping project activities against a state-of-the-art software process reference model gave additional confidence for monitoring the OMEGA system test process. An indication of the added confidence was that typically an existing metric was linked to new, capability related questions. The continuous assessment information also provided new insights in the GQM feedback sessions. For example, the factors impacting actual process execution became very clear for the process participants, this thereby resulting in an improved process execution.

The GQM feedback session was found valuable also from the MCA point of view, providing a good feedback mechanism for evaluating the MCA results. In addition, the potential to reuse metrics and their definitions within a measurement program and for other projects was seen as a positive finding. Finally, the effective use of checklists indicates that MCA may be used also with a relatively light technological infrastructure for measurement data collection.

6. Summary and Conclusions

6.1 Research results and contributions

Software process assessment and improvement are interdependent approaches that are successfully used by an increasing number of organisations. This research has been motivated by the shortcomings of the current assessment approaches to provide enhanced support for process improvement.

The research problem was

- **How can an industrial organisation monitor the status of its software process using measurement based continuous assessments?**

The hypothesis was

- **Measurements may be successfully embedded into the software process to support regular assessment.**

Especially interesting for this research was to understand and improve the way an organisation can monitor the software process status between regular assessments and how this monitoring can be achieved in an industrial setting.

To prove the hypothesis and resolve the research problem an operational method for measurement based continuous assessment (MCA) was developed to complement existing assessment approaches. The method for MCA was applied in two case organisations that considered the approach both feasible and useful. The Tokheim case was included in Chapter 5 of this thesis. The Dräger MT-M case is described in the internal PROFES documents – "Continuous assessment is feasible and it takes little time to do an assessment. Dräger MT-M therefore will continue with this way of process assessment" (van Uijtregt 1999). A cost model was built to understand the costs related to the application of MCA in the Tokheim case, and the costs were considered reasonable. Three support tools were built and much interest was invested in studying the tool support. However, while tool support was considered important there are several effective techniques for doing MCA even with limited tool support.

The MCA method has been integrated into the PROFES improvement methodology (PROFES-Consortium 2000) and the FAME assessment method (Beitz & Järvinen 2000). The research has been performed in both academic and industrial settings. The author has had a central role in defining the concepts and making the measurement based continuous assessment operational. Furthermore, the author has defined most of the initial requirements for the tools and was responsible for the methodology development in the PROFES project, and was the co-author of the FAME methodology. The specific role of the author in each of the original papers included in this thesis is discussed in chapter 7.

6.2 Answers to the research questions

The answers to the research questions defined in Chapter 1.3 are concluded as follows:

How does continuous assessment differ from other assessments?

The characteristics of continuous assessment were developed by examining the current assessment approaches based on the hypothesis of this thesis. In short, what seems to be lacking is flexibility for different situations, and the use of common frameworks to better understand, manage and utilise measurements. This resulted in the assessment typology in Chapter 3 and MCA definition in Chapter 4 to show how continuous assessment can augment other assessments.

What techniques and support are needed for establishing measurement based continuous assessment?

The techniques and support for MCA were developed based on the MCA definition in Chapter 4.1. These are documented in Chapters 4.2, 4.3 and 4.4. In summary, the method presented in Chapter 4.2 provides the basis for establishing MCA. Evaluation of the techniques and tool support in the industrial case presented in Chapter 5 resulted in conclusion that MCA may also be possible with limited tool support and process capability.

Is it feasible to use measurement based continuous assessment in an industrial setting?

The approach for MCA was applied in an industrial setting as documented in Chapter 5. The two case organisations – the other with previous measurement expertise, the other not – considered the approach both feasible and useful. A cost model was built to understand the costs related to the application of MCA in the Tokheim case where the costs were considered reasonable. More cases would of course have been preferable, but even so the results seem valuable and transferable to other companies as the case organisations were involved in the development of MCA for over a year participating in numerous meetings and reviews both on technical and managerial level.

6.3 Recommendations for future research

This thesis has examined continuous assessment and especially measurement based continuous assessment (MCA) of software engineering processes mainly under the auspices of the PROFES project. While there is some confidence in the developed approach for MCA as it was developed in close co-operation with the industrial partners of PROFES, MCA still needs to be taken on trial in multiple instances and different contexts in order for it to be more generally validated. The use of other reference frameworks than ISO 15504, such as CMMI or ISO 9001, should be tested with the MCA approach. The costs and benefits of using MCA need to be collected from these trials, and analysed to build a better understanding on the risks and applicability of the approach. Conceptually, one of the strengths of MCA seems to be that it is flexible enough to be used to complement any reference framework. Hence, MCA can probably be used to support more general measurement frameworks, such as BITS (2000). A practical research effort helping to lower the threshold of MCA use for companies would be to gather and package a set (or sets) of assessment indicators that could be used as a basis for MCA. This is a major task, however, which was attempted at VTT Electronics already in 1996 in the PROAM project (Parviainen et al. 1996) with moderate success due to limited resources and the vast number of possible combinations.

There remain many other challenges for the MCA. The promise of the “super metrics” as discussed in Chapter 4.1.4 on page 50, is perhaps the most interesting future research direction, but better tool support and non-technical aspects of assessment and measurement also are inviting areas of inquiry. For example, how to integrate MCA more effectively with traditional assessment approaches and the measurement process of an organisation? Further, there is the impact of process and measurement maturity on assessment, which should be investigated. It seems that in very advanced organisations it is certainly possible to use MCA or a similar approach for the assessment, but does the reverse also hold? Is MCA too difficult to implement in very low maturity organisations, and if so, are there favourable conditions, such as tool support, which can help alleviate the problems? Lastly, it would seem that MCA could be used with product/process dependency (PPD) models. The PPD models were investigated in the PROFES project as a core component for enabling product focused process improvement (Hamann et al. 1998). Augmenting PPD models with assessment indicators and using MCA for assessing PPD suitability and validation could add some new value to the product driven software process improvement.

Finally, it is obvious that the very foundations of the software process paradigm need to be investigated. It is generally agreed that there is no sound basis for example for the concepts of software process maturity and capability, except for some loose analogy to statistical process control, which remains largely unvalidated. The state of matters reflects the youth of software process research, but should not serve as an excuse to overlook building a solid theory for this promising field within software engineering.

7. Introduction to the papers

This chapter gives an overview of the original publications included in this thesis. The content of each paper is briefly discussed in the following sections. The author of this dissertation thesis is the principal author in papers I (authors in alphabetical order), III, V and VIII. In papers II, IV, VI and VII, the contribution and effort of the author of this dissertation has also been essential. Most papers have been written with multiple authors due to the nature of the related research projects.

7.1 Paper I, Multidimensional approach to IS quality

Dahlberg, T. & Järvinen, J. 1997. Challenges to IS Quality. Information and Software Technology Journal, Vol. 39, No.12, pp. 809 - 818.

Paper I characterises the various factors related to quality in information systems and technology. The paper summarises the landscape of IS quality that is the basis for further research presented in this thesis.

The paper contains a description of:

- TQM and related approaches
- Three dimensions of quality: technical, organisational and use quality
- Discussion on challenges to IS quality.

The authors in this paper are listed in alphabetical order due to their equal effort for the paper.

7.2 Paper II, Experiences of using MetriFlame

Parviainen, P., Järvinen, J. & Sandelin, T. 1997. Practical Experiences of Tool Support in a GQM-based Measurement Programme. Software Quality Journal, Vol. 6, No. 4, pp. 283 - 294.

Paper II presents experiences of supporting a software measurement program with MetriFlame tool environment. MetriFlame was originally created with the intention to support the MCA approach – or automated assessment, as the term was at that time. The experiences with MetriFlame have been extremely valuable for the design process of MCA from early on. Many MCA essentials in their early form are to be found already in this paper.

The paper contains a description of:

- MetriFlame tool environment
- Case study for using MetriFlame in an industrial organisation
- Experiences of tool support for measurement program.

The author of this thesis was a co-author and the key contributor to the MetriFlame concepts and design described in this paper.

7.3 Paper III, Principles of using SPICE as a measurement tool

Järvinen, J. 1998. Facilitating Process Assessment with Tool Supported Measurement Programme. In: Coombes, H., Hooft van Huysduynen, M., Peeters, B. (eds.), Proceedings of FESMA98 – Business Improvement Through Software Measurement. Technological Institute, Antwerp, Belgium, May 6 - 8, pp. 606 - 614.

With paper III, the basic idea of using a process assessment reference framework as a measurement instrument is introduced. The paper still revolves around the assumption that tools are an absolute prerequisite for establishing MCA. This assumption – as the only option – was later discarded. The paper contains:

- Idea of embedding measurements in software engineering work
- Description of how integration between SPICE and GQM can work in principle
- Discussion of the required tool support for MCA.

The author of this thesis was the principal author and contributor of this paper.

7.4 Paper IV, Experiences of using GQM in an industrial setting

Birk, A., van Solingen, R., & Järvinen, J. 1998. Business Impact, Benefit, and Cost of Applying GQM in Industry: An In-Depth, Long-Term Investigation at Schlumberger RPS. In: Proceedings of the Fifth International Symposium on Software Metrics (METRICS´98). Bethesda, Maryland, November 20 - 21, pp. 93 - 96.

Paper IV reports experiences and lessons learnt from an industrial application of GQM. The paper addresses benefits and costs of GQM measurement and identifies success factors for GQM use. This paper helps to understand GQM from a practical perspective and builds a basis for planning how to apply MCA as many of the findings, such as the GQM success factors, are expected to be valid also for MCA.

The paper contains a summary of:

- Benefits from GQM measurement
- Cost of GQM measurement
- Success factors for GQM measurement
- Discussion of operational GQM enhancements.

The author of this thesis did not participate in the fieldwork for this paper but he had a major role in the analysis of the findings.

7.5 Paper V, Establishing MCA

Järvinen, J. & van Solingen, R. 1999. Establishing Continuous Assessment Using Measurements. In: Proceedings of the 1st International Conference on Product Focused Software Process Improvement (PROFES´99). Oulu, Finland, June 22 - 24, pp. 49 - 67.

Paper V presents the principles of measurement based continuous assessment. It also outlines a practical approach for establishing MCA for software engineering processes – or MAA as the term was at that time. This paper was a result of the exploratory work done with MCA in the PROFES project to validate the MCA concepts and contrive the method.

The paper describes the:

- Principles of MCA
- Preliminary method for establishing MCA
- Preliminary MCA experiences from PROFES.

The author of this thesis was the principal author and contributor of this paper.

7.6 Paper VI, Integration of assessment and measurement

Vierimaa, M., Hamann, D., Komi-Sirviö, S., Birk, A., Järvinen, J. & Kuvaja, P. 1999. Integrated Use of Software Assessments and Measurements. In: Proceedings of the 11th International Conference on Software Engineering and Knowledge Engineering (SEKE '99). Kaiserslautern, Germany, June 17 - 19, pp. 83 - 87.

Paper VI discusses the potential synergies of using software process assessment and measurement together in the various stages of improvement work. The integration was studied in the PROFES project during actual assessments and measurement programs. This paper helps to position MCA and forms an important link and interface between MCA and assessment and measurement activities in general.

The paper contains:

- Possibilities of assessment and measurement integration
- Preliminary integration experiences from PROFES.

The author of this thesis was a co-author and participated actively in the development of the concepts for this paper, and he was responsible for the methodology development. Hence, the multitude of authors in this paper reflects the defined writing procedure in the PROFES research project.

7.7 Paper VII, Role of measurement in a modern improvement methodology

Hamann, D., Pfahl, D., Järvinen, J. & van Solingen, R. 1999. The Role of GQM in the PROFES Improvement Methodology. In: Proceedings of the 3rd International Conference on Quality Engineering in Software Technology (CONQUEST '99). Nürnberg, Germany, September 26 - 27, pp. 64 - 79.

Paper VII defines the role of GQM in the PROFES improvement methodology using practical examples. There are many facets of measurement that can be applied within SPI, which are integrated into the PROFES improvement methodology. This paper shows the relationships between GQM and PROFES and describes how MCA fits in.

The paper contains:

- Outline of the PROFES improvement methodology
- Principles of GQM
- Roles of GQM in PROFES.

The author of this thesis was a co-author and participated actively in the development of the concepts for this paper, and he was the responsible for the methodology development in the PROFES project.

7.8 Paper VIII, MCA in practice

Järvinen, J., Hamann, D. & van Solingen, R. 1999. On Integrating Assessment and Measurement: Towards Continuous Assessment of Software Engineering Processes. In: Proceedings of the Sixth International Symposium on Software Metrics (METRICS´99). Boca Raton, Florida, November 4 - 6, pp. 22 - 30.

Paper VIII summarises the MCA principles and presents a case study of using MCA in an industrial organisation. The case study validates the MCA concepts and introduces practical techniques and tips. MCA feasibility is also discussed and a cost model for its application is presented.

The paper contains:

- Summary of MCA principles
- Case study of applying MCA in an industrial organisation
- MCA cost model.

The author of this thesis was the principal author and contributor of this paper.

References

Abdel-Hamid, T. & Madnick, S. E. 1991. Software Project Dynamics: An Integrated Approach. Englewood Cliffs, New Jersey. Prentice-Hall. 288 p.

Bach, J. 1994. The Immaturity of the CMM. The American Programmer. Vol. 7. September. Pp. 13 - 18.

Bach, J. 1995. Enough About Process: What We Need Are Heroes. IEEE Software. Vol. 12. No. 2. Pp. 96 - 98.

Bandinelli, S., Fuggetta, A., Lavazza, L., Loi, M. & Picco, G. P. 1995. Modeling and Improving an Industrial Software Process. IEEE Transactions on Software Engineering. Vol. 21. No. 5 May. Pp. 440 - 453.

Barker, H., Dorling, A. & Simms, P. G. 1992. The ImproveIT Project. European Conference on Software Quality. Madrid 3 - 6.11. 12 p.

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

Basili, V. R., Caldiera, G. & Rombach, H. D. 1994a. Experience Factory. Encyclopaedia of Software Engineering . Volume 1. Pp. 469 - 476.

Basili, V. R., Caldiera, G. & Rombach, H. D. 1994b. Goal Question Metric Paradigm. Encyclopaedia of Software Engineering . Volume 1. Pp. 528 - 532.

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. R. & Weiss, D. M. 1984. A methodology for collecting valid software engineering data. IEEE Transactions on Software Engineering. Vol. 10. No. 6. Pp. 728 - 738.

Bassman, M., McGarry, J. F. & Pajerski, R. 1994. Software Measurement Guidebook. SEL-94-102. 148 p.

Baumert, J. H. & McWhinney, M. S. 1992. Software Measures and the Capability Maturity Model. Pittsburgh, PA. Software Engineering Institute at Carnegie Mellon University. CMU/SEI-92-TR-25. 306 p.

Beitz, A., El-Emam, K. & Järvinen, J. 1999. A Business Focus to Assessments. In the Proceedings of SPI´99. Barcelona, Spain. 29 p.

Beitz, A. & Järvinen, J. 2000. FAME - an Approach for Software Process Assessment. Kaiserslautern. Fraunhofer Institute for Experimental Software Engineering. No. 001.00/E. 72 p.

Bergmann, J. 1999. EFQM and BOOTSTRAP. In the Proceedings of 2nd BOOTSTRAP Assessor Day. Cologne, Germany. The BOOTSTRAP Institute. 18 p.

Bicego, A., Khurana, M. & Kuvaja, P. 1998. BOOTSTRAP 3.0 - Software Process Assessment Methodology. In the Proceedings of the SQM '98. 13 p.

Birk, A., Giese, P., Kempkens, R., Rombach, D. & Ruhe, G., Eds. 1997. The PERFECT Handbook. Fraunhofer IESE Reports Nr. 059.97 - 062.97. Kaiserslautern, Fraunhofer Einrichtung für Experimentelles Software Engineering.

Birk, A., van Solingen, R. & Järvinen, J. 1998. Business Impact, Benefit, and Cost of Applying GQM in Industry: An In-Depth, Long-Term Investigation at Schlumberger RPS. In the Proceedings of the 5th International Symposium on Software Metrics (Metrics '98). Bethesda, Maryland, USA. IEEE Computer Society Press. 4 p.

BITS 2000. BITS - Primer: an Executive Seminar on the Balanced IT Scorecard. ESI, Spain. ESI-2000-BITS-PRIMER-V1.0. 119 p.

Bollinger, T. B. & McGowan, C. 1991. A Critical Look at Software Capability Evaluations. IEEE Software. Vol. 8. July. Pp. 25 - 41.

Bourque, P., Dupuis, R., Abran, A., Moore, J. W. & Tripp, L. 2000. SWEBOK - Guide to the Software Engineering Body of Knowledge – A Stone Man Version. Dépt. d'Informatique, UQAM. Available from http://www.swebok.org/stoneman/version06.html.

Braa, K. 1995. Beyond Formal Quality in Information Systems Design - A Framework for Information Systems Design from a Quality Perspective. In the Proceedings of IRIS 18. Gothenburg. Gothenburg University, Studies in Informatics, Report 7. Pp. 97 - 113.

Briand, L., Differding, C. & Rombach, H. D. 1996. Practical guidelines for measurement-based process improvement. Software Process Improvement & Practice. December 2(4). Pp. 253 - 280.

Briand, L., El Emam, K. & Melo, W. L. 1999. An Inductive Method fo Software Process Improvement: Concrete Steps and Guidelines. Elements of Software Process Assessment and Improvement. K. El Emam & N. H. Madhavji (eds.). Los Alamitos, CA. IEEE Computer Society. Pp. 113 - 130.

Brooks, F. J. 1987. No Silver Bullet: Essence and Accidents of Software Engineering. IEEE Computer. Vol. 20. No. 4. Pp. 10 - 19.

Byrnes, P. & Phillips, M. 1996. Software Capability Evaluation, Version 3.0. Pittsburgh, PA. Software Engineering Institute. CMU/SEI-96-TR-002. 192 p.

Campbell, M. 1995. Tool Support for Software Process Improvement and Capability Determination: Changing the Paradigm of Assessment. Software Process Newsletter. No. 4. Fall. Pp. 12 - 15.

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

Card, D. 1992. Capability Evaluations Rated Highly Variable. IEEE Software. September. Pp. 105 - 106.

Christie, A. M. 1994. Software Process Automation. Berlin. Springer-Verlag. 215 p.

Clark, B. K. 1997. The Effects Of Software Process Maturity On Software Development Effort. University of Southern California. 140 p.

CMMI 2000. CMMI-SE/SW, V1.0 Capability Maturity Model ® – Integrated for Systems Engineering/Software Engineering, Version 1.0 Continuous Representation. Pittsburgh, PA. Software Engineering Institute. CMU/SEI-2000-TR-019. 618 p.

Crosby, P. B. 1979. Quality is Free. McGraw-Hill. 270 p.

Cugola, G. & Ghezzi, C. 1998. Software processes: a Retrospective and a Path to the Future. Software Process - Improvement and Practice. Vol. 4. No. 3. Pp. 101 - 123.

Curtis, B. 2000. The Cascading Benefits of Software Process Improvement (Keynote). PROFES 2000. Oulu, Finland. Springer-Verlag. 21 p.

Curtis, B., Hefley, W. E. & Miller, S. 1995. Overview of the People Capability Maturity Model. CMU/SEI-95-MM-01. 77 p.

Curtis, B., Kellner, M. & Over, J. 1992. Process Modelling. Communications of the ACM. Vol. 35. No. 9 Sept. Pp. 75 - 90.

Davenport, T. H., Jarvenpaa, S. L. & Beers, M. C. 1996. Improving Knowledge Work Process. Sloan Management Review. Summer. Pp. 53 - 65.

DeMarco, T. 1982. Controlling Software Projects: Management, Measurement and Estimation. Prentice-Hall. 284 p.

DeMarco, T. & Lister, T. 1999. Peopleware: Productive Projects and Teams. Dorset House. 264 p.

Deming, W. E. 1986. Out of the Crisis: Quality, Productivity and Competitive Position. Cambridge. Mass.: MIT Center for Advanced Engineering Study. 507 p.

Deutsch, M. S. 1992. Total Quality Management in Hughes Aircraft. The Esprit Bootstrap Conference on Lean Software Development. Stuttgart, 22. - 23.10. Steinbeis-Zentrum, Europäischer Technologietransfer, Stuttgart. 41 p.

Dion, R. 1993. Process Improvement and the Corporate Balance Sheet. IEEE Software. October. Pp. 28 - 35.

DOD-STD-2167A 1988. Military Standard DOD-STD-2167A Defence System Software Development. Department of Defence. USA. 50 p.

Doiz, I. 1997. ESI News - BootCheck. Software Process Improvement & Practice. Vol. 3. No. 1. Pp. 62 - 63.

Dorling, A. & Simms, P. 1992. Study Report: The Need and Requirements for a Software Process Assessment Standard. International Standards Organization. ISO/IEC JTC1/SC7/N944R.

Drouin, J.-N. 1999. The SPICE project. Elements of Software Process Assessment and Improvement. K. El Emam & N. H. Madhavji (eds.). Los Alamitos, CA. IEEE Computer Society. Pp. 45 - 55.

Dunaway, D. K. & Masters, S. 1996. CMM -Based Appraisal for Internal Process Improvement (CBA IPI): Method Description. Pittsburgh, PA. Software Engineering Institute. CMU/SEI-96-TR-007. 57 p.

Dunaway, D. K., Seow, M. L. & Baker, M. 2000. Analysis of Lead Assessor Feedback for CBA IPI Assessments Conducted July 1998 - October 1999. CMU/SEI-2000-TR-005. 50 p.

Dutta, S., van Wassenhove, L., Rementeria, S. & Doiz, I. 1996. 1995/1996 Software Excellence Survey: Model and Detailed Results Analysis. European Software Institute. ESI-1996-PIA/96282. 57 p.

Earthy, J. 2000. Usability Maturity Model: Processes. INUSE. Available from http://www.lboro.ac.uk/research/husat/eusc/.

EFQM 1999. Introducing Excellence. European Foundation for Quality Management. Available from
http://www.efqm.org/members/info/1312-InEx-en-bw.pdf.

El Emam, K. 1998. The Internal Consistency of the ISO/IEC 15504 Software Process Capability Scale. Kaiserslautern. Fraunhofer IESE. ISERN-98-06. 13 p.

El Emam, K. & Goldenson, D., R. 1999. An Empirical Review of Software Process Assessments. Ottawa. Institute for Information Technology. National Research Council Canada. ERB-1065. 84 p.

Eriksson, I. & Törn, A. 1991. A Model for IS Quality. Software Engineering Journal. Vol. 6. July. Pp. 152 - 158.

ESA 1991. ESA PSS-05-0 Software Engineering Standards. European Space Agency. Issue 2. 130 p.

Etnoteam 1998. Capability Trend Analysis Tool Etnoteam. Milan, Italy. CD-ROM.

Fenton, N. E. & Pfleeger, S. L. 1996. Software Metrics - A Practical and Rigorous Approach. Thomson Computer Press. 638 p.

Ferguson, P. 1999. Software process improvement works! : Advanced Information Services Inc. Pittsburgh, Pa. Carnegie Mellon University Software Engineering Institute. CMU/SEI-99-TR-027. 36 p.

Fisher, M. 1998. Software acquisition improvement framework (SAIF) definition. Pittsburgh, PA. Carnegie Mellon University Software Engineering Institute. CMU/SEI-98-TR-003. 36 p.

Florac, W. A. & Carleton, A. D. 1999. Measuring the software process : statistical process control for software process improvement. Reading, Mass. Addison-Wesley. 250 p.

Gilb, T. 1992. Quality Planning - Result Planware. The Esprit Bootstrap Conference on Lean Software Development. Stuttgart, 22. - 23.10. Steinbeis-Zentrum, Europäischer Technologietransfer, Stuttgart. 64 p.

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

Grady, R. B. & Caswell, D. L. 1987. Software Metrics: Establishing a Company-Wide Program. Englewood Cliffs. Prentice-Hall. 288 p.

Gryna, F. M. & Juran, J. M. 1980. Quality Planning and Analysis. From Product Development through Use. New York, NY. McGraw-Hill. 634 p.

Hamann, D., Järvinen, J., Birk, A. & Pfahl, D. 1998. A Product-Process Dependency Definition Method. The 24th EUROMICRO Conference, Workshop on Software Process and Product Improvement. Västerås, Sweden. IEEE Computer Society Press. Pp. 898 - 904.

Herbsleb, J., Carleton, A., Rozum, J., Siegel, J. & Zubrow, D. 1994. Benefits of CMM-Based Software Process Improvement: Initial Results. Software Engineering Institute, Carnegie Mellon University. CMU/SEI-94-TR-13. 64 p.

Herbsleb, J., Zubrow, D., Goldenson, D., Hayes, W. & Paulk, M. 1997. Software Quality and the Capability Maturity Model. Communications of the ACM. June 1997. Pp. 30 - 40.

Hsia, P. 1996. Making Software Development Visible. IEEE Software. March. Pp. 23 - 25.

Humphrey, W. S. 1987. Characterizing the Software Process: A Maturity Framework. Software Engineering Institute. CMU/SEI-87-TR-11. Also published in IEEE Software, Vol. 5, No. 2, March 1988. Pp. 73 - 79 p.

Humphrey, W. S. 1989. Managing the Software Process. Addison-Wesley. 494 p.

Humphrey, W. S. 1991. Comments on 'A Critical Look'. IEEE Software. July. Pp. 42 - 46.

Humphrey, W. S. 1997. Introduction to the Personal Software Process. Addison-Wesley. 304 p.

Humphrey, W. S. 1999. Competing in the Software Age. Software Engineering Institute. Available from http://www.sei.cmu.edu/videos/watts/DPWatts.mov.

Humphrey, W. S. 2000. Introduction to the Team Software Process. Addison-Wesley. 496 p.

Humphrey, W. S., Snyder, T. R. & Willis, R. R. 1991. Software Process Improvement at Hughes Aircraft. IEEE Software. July. Pp. 11 - 23.

IEC-61508-3 1998. Functional safety of electrical/electronic/programmable electronic safety-related systems - Part 3: Software requirements. IEC. Geneva, Switzerland. 95 p.

Ishikawa, K. 1985. What is Total Quality Control? The Japanese Way. New Jersey. Prentice-Hall. 240 p.

ISO/IEC-8402 1994. Quality management and quality assurance - Vocabulary. International Organisation for Standardisation (Ed.). 48 p.

ISO/IEC-9000-3 1997. Guidelines for the application of ISO 9001 to develop, supply, install and maintain software. International Standards Organization (Ed.). 34 p.

ISO/IEC-9001 1994. Quality Systems; Model for Quality Assurance in Design/Development, Production, Installation and Servicing. International Standards Organization (Ed.). 31 p.

ISO/IEC-9126 1991. Information technology - Software product evaluation - Quality characteristics and guidelines for their use. International Organisation for Standardisation (Ed.). CH-1211 Geneva, Switzerland, Casa Postale 56. 13 p.

ISO/IEC-12207 1995. Information Technology - Software lifecycle process. International Standards Organization. 68 p.

ISO/IEC-15504-2 1998. Information Technology - Software Process Assessment - Part 2: A Reference Model for Processes and Process Capability. International Organisation for Standardisation (Ed.). CH-1211 Geneva, Switzerland, Casa Postale 56. 44 p.

ISO/IEC-15504-5 1998. InformationTechnology - Software Process Assessment - Part 5: An Assessment Model and Indicator Guidance. International Organisation for Standardisation (Ed.). CH-1211 Geneva, Switzerland, Casa Postale 56. 128 p.

ISO/IEC-15504-7 1998. Information technology - Software process assessment - Part 7: Guide for use in process improvement. International Organisation for Standardisation (Ed.). CH-1211 Geneva, Switzerland, Casa Postale 56. 41 p.

Johnson, D. L. & Brodman, J. G. 1999. Tailoring the CMM for Small Businesses, Small Organizations, and Small Projects. Elements of Software Process Assessment and Improvement. K. El Emam & N. H. Madhavji (eds.). Los Alamitos, CA. IEEE Computer Society. Pp. 237 - 257.

Jones, C. 1999. The Economics of Software Process Improvements. Elements of software process assessment and improvement. K. El Emam & N. H. Madhavji (eds.). Los Alamitos, Calif. IEEE Computer Society. Pp. 133 - 150.

Järvinen, J. 1994a. BOOTSTRAP: Improving the Capability of Software Industry with Database Support. In the Proceedings of the First IFIP/SQI International Conference on Software Quality and Productivity. Hong Kong. 8 p.

Järvinen, J. 1994b. On Comparing Process Assessment Results: BOOTSTRAP and CMM. In the Proceedings of Second International Conference on Software Quality Management. Edinburgh, Scotland, UK. 16 p.

Järvinen, J., Komi-Sirviö, S. & Ruhe, G. 2000. The PROFES Improvement Methodology: Enabling Technologies and Methodology Design. In the Proceedings of PROFES 2000 Conference. Oulu, Finland. Springer-Verlag. Pp. 257 - 270.

Järvinen, P. 1999. On Research Methods. Tampere, Finland. Opinpaja Oy. 129 p.

Kaplan, R. S. & Norton, D. P. 1996. The Balanced Scorecard: Translating Strategy into Action. Harvard Business School Press. 322 p.

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.

Kellner, M. & Hansen, G. 1988. Software Process Modeling. Software Engineering Institute. CMU/SEI-88-TR-9. 58 p.

Kempkens, R., Rösch, P., Scott, L. & Zettel, J. 2000. Instrumenting Measurement Programs with Tools. In the Proceedings of PROFES 2000 Conference. Oulu, Finland. Springer-Verlag. Pp. 353 - 375.

Kenett, R. & Zacks, S. 1998. Modern Industrial Statistics. Belmont, CA. Duxbury Press. 621 p.

Kinnula, A. 1999. Software Process Engineering in a Multi-Site Environment: An Architectural Design of a Software Process Engineering System. Oulu, Finland. Oulu University Press. 119 p.

Kitchenham, B. 1996. Software Metrics: Measurement for Software Process Improvement. Cambridge, Mass. Blackwell. 241 p.

Krasner, H. 1999. The Payoff for Software Process Improvement: What it is and How to Get it. Elements of software process assessment and improvement. K. El Emam & N. H. Madhavji (eds.). Los Alamitos, Calif. IEEE Computer Society. Pp. 151 - 176.

Kuvaja, P., Similä, J., Krzanik, L., Bicego, A., Saukkonen, S. & Koch, G. 1994. Software Process Assessment & Improvement - The BOOTSTRAP Approach. Oxford, UK. Blackwell Publishers. 149 p.

Lillrank, P. 1990. Laatumaa (Land of Quality). Helsinki. Gaudeamus. 277 p.

March, S. T. & Smith, G. F. 1995. Design and Natural Science Research on Information Technology. Decision Support Systems. No. 15. Pp. 251 - 266.

Masters, S. & Bothwell, C. 1995. CMM Appraisal Framework. Software Engineering Institute. CMU/SEI-95-TR-001. 76 p.

MBNQA 2000. Malcolm Baldridge National Quality Award: Application Guidelines. National Institute of Standards and Technology. Available from http://www.quality.nist.gov/apps_forms_instr.htm.

McFeeley, B. 1996. IDEAL: A User´s Guide for Software Process Improvement. Carnegie Mellon University, Software Engineering Institute. CMU/SEI-96-HB-001. 236 p.

Miyazaki, Y., Ohtaka, Y., Kubono, K., Fujino, A. & Muronaka, K. 1995. Software Process Assessment and Improvement based on Capability Maturity Model for Software. In the Proceedings of the First World Congress for Software Quality (WCSQ). San Francisco. 16 p.

Mosemann, I. 1994. Let´s Write Finis to the Black Hole Syndrome. Crosstalk. Available from http://stsc.hill.af.mil/CrossTalk/1994/oct/xt94d10a.asp.

Nevalainen, R. 2000. Travel report from SC7 meeting. Espoo, Finland. STTF. Fisma. 11 p.

Niessink, F. & van Vliet, H. 1999. IT Service Capability Maturity Model. Amsterdam. Vrije Universiteit Amsterdam. IR-463, Release L2-1.0.

Nonaka, I. & Takeuchi, H. 1995. The knowledge-creating company: How Japanese companies create the dynamics of innovation. New York. Oxford University Press. 284 p.

Park, R. E. 1996. CMM Version 1.1 measurement map. Pittsburgh, PA. Software Engineering Institute at Carnegie Mellon University. CMU/SEI-96-SR-003. 53 p.

Park, R. E., Goethert, W. B. & Florac, W. A. 1996. Goal-Driven Software Measurement — A Guidebook. Pittsburgh, PA. Software Engineering Institute at Carnegie Mellon University. CMU/SEI-96-HB-002. 189 p.

Parviainen, P., Sandelin, T. & Järvinen, J. 1996. Automating the Data Collection of the SPICE Practices: A Classification Framework. Oulu, Finland. VTT Electronics. Internal PROAM project report. 65 p.

Paulk, M. C., Curtis, B., Chrissis, M. B. & Weber, C. 1993. Capability Maturity Model for Software, Version 1.1. Software Engineering Institute. CMU/SEI-93-TR24. 82 p.

Paulk, M. C., Goldenson, D. & White, D. M. 2000. The 1999 Survey of High Maturity Organizations. Software Engineering Institute. CMU/SEI-2000-SR-002. 95 p.

Pfleeger, S. L. 1998. Understanding and Improving Technology Transfer in Software Engineering. DACS. DACS-SOAR-98-1. 23 p.

Pierce, B. 2000. Is CMMI ready for Prime Time? Crosstalk. Available from http://stsc.hill.af.mil/crosstalk/2000/jul/pierce.asp.

PROAM 1996. PROAM Strategic Project - Internal documents VTT Electronics. Oulu. CD-ROM.

PROFES-Consortium 2000. The PROFES User Manual. Stuttgart. Fraunhofer IRB Verlag, Germany. 400 p.

PROFES-Internal 1997 - 1999. Internal documents VTT Electronics. Oulu. CD-ROM.

Pulford, K., Kuntzmann-Combelles, A. & Shirlaw, S. 1996. A Quantitative Approach to Software Management - The ami Handbook. Addison-Wesley. 179 p.

Pyzdek, T. 1992. To Improve Your Process: Keep It Simple. IEEE Software. September. Pp. 112 - 113.

Radice, R. A., Harding, J. T., Munnis, P. E. & Phillips, R. W. 1985. A Programming Process Study. IBM Systems Journal. Vol. 24. No. 2. Pp. 91 - 101.

Rombach, D. 2000. Capitalizing on Experience (Keynote). PROFES 2000. Oulu, Finland. Springer-Verlag. 20 p.

SEMA 2000. Process Maturity Profile of the Software Community 1999 Year End Update. Software Engineering Institute at Carnegie Mellon University. Available from http://www.sei.cmu.edu/sema/pdf/2000mar.pdf.

Sheard, S. A. 1997. The Frameworks Quagmire. Crosstalk. Available from http://www.stsc.hill.af.mil/crosstalk/1997/sep/frameworks.asp.

Shewhart, W. A. 1939. Statistical Method from the Viewpoint of Quality Control, Washington: Graduate School of Agriculture. Referenced in Deming, W. E. 1986, Out of the Crisis. Cambridge, Mass.: MIT Center for Advanced Engineering Study. 507 p.

Simmons, D. B., Ellis, N. C., Fujihara, H. & Kuo, W. 1998. Software Measurement - A Visualization Toolkit for Project Control and Process Improvement. Upper Saddle River, NJ. Prentice Hall. 442 p.

SPICE-4 1995. ISO/IEC Software Process Assessment Part 4: Guide to conducting assessment. ISO/IEC/ JTC1/SC7/WG10. Doc. Ref. 7N1408. 33 p.

SPICE-5 1995. ISO/IEC Software Process Assessment Part 5: Construction, selection and use of assessment instruments and tools. ISO/IEC/JTC1/SC7/ WG10. Doc. Ref. 7N1409. 132 p.

SPICE 1998. Phase 2 Trials Interim Report. SPICE Project Trials Team. Available from http://www.iese.fhg.de/SPICE/Trials/p2rp100pub.pdf.

Steinmann, C. & Stienen, H. 1996. SynQuest - Tool Support for Software Self-Assessments. Software Process - Improvement and Practice. Vol 2. No. 1. Pp. 5 - 12.

Trochim, W. 1997. The Research Knowledge Base. Cornell University. Available from http://trochim.human.cornell.edu/kb/.

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. Vol. 15. No. 1. Pp. 78 - 86.

van Solingen, R. 2000. Product Focused Software Process Improvement: SPI in the embedded software domain. Eindhoven, the Netherlands. Technische Universiteit Eindhoven. 192 p.

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

van Uijtregt, A. 1999. Experience Package Dräger MT-M. Esprit #23232 PROFES. PROFES-2.5.B.4-II. 19 p.

van Veldhoven, N. J. 1999. Continuous Assessment: Combining Software Process Assessment and Goal-oriented Measurement. Eindhoven. Eindhoven University of Technology. 60 p.

Weinberg, G. M. 1992. Quality Software Management - Volume I: Systems Thinking. New York, NY. Dorset House. 336 p.

Whitney, R., Nawrocki, E. M., Hayes, W. & Siegel, J. 1994. Interim Profile: Development and Trial of a Method to Rapidly Measure Software Engineering Maturity Status. Software Engineering Institute. CMU/SEI-94-TR-4. 44 p.

Willis, R. R. 1998. Hughes aircraft's widespread deployment of a continuously improving software process. Pittsburgh, Pa. Carnegie Mellon University Software Engineering Institute. CMU/SEI-98-TR-006. 86 p.

VTT 1999. MetriFlame - A Measurement and Feedback Tool Environment (v. 1.3). VTT Electronics. Oulu, Finland. CD-ROM.

VTT 2000. SPICE Mapper (v. 1.0). VTT Electronics. Oulu, Finland. CD-ROM.

Yin, R. K. 1991. Case study research: design and methods. Sage Publications. 165 p.

Zubrow, D. 1997. The Evolution of Measurement with CMM -based Software Process Improvement. Software Engineering Symposium. Software Engineering Institute at Carnegie Mellon University, Pittsburgh, PA. 28 p.

Zultner, R. E. 1990. Software Total Quality Management (TQM): What Does it Take to be World Class? American Programmer. Vol. 3. No. 11. Pp. 2 - 11.

Appendices of this publication are not included in the PDF version. Please order the printed version to get the complete publication (http://otatrip.hut.fi/vtt/jure/index.html)

Published by

VTT

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 426
VTT–PUBS–426

Author(s)

Järvinen, Janne

Title

Measurement based continuous assessment of software engineering processes

Abstract

Software process assessments are routinely used by the software industry to evaluate software processes before instigating improvement actions. They are also used to assess the capability of an organisation to produce software. Since assessments are perceived as expensive, time-consuming and disruptive for the workplace there is a need to find alternative practices for software process assessment. Especially interesting for this research was to understand and improve the way an organisation can monitor the software process status between regular assessments and how this monitoring can be achieved feasibly in an industrial setting.

This thesis proposes a complementary paradigm for software process assessment – measurement based continuous assessment. This approach combines goal-oriented measurement and an emerging standard for software process assessment as the background framework for continuous assessment of the software engineering process. Software tools have been created to support the approach that has been tested in an industrial setting. The results show that the proposed approach is feasible and useful, and provides new possibilities and insights for software process assessment.

Keywords

software engineering, software process, software process assessment, software measurement, software process improvement

Activity unit

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

ISBN 951–38–5592–9 (soft back ed.) 951–38–5593–7 (URL: http://www.inf.vtt.fi/pdf/)			Project number
Date November 2000	Language English	Pages 97 p. + app. 90 p.	Price D
Name of project		Commissioned by	
Series title and ISSN VTT Publications 1235–0621 (soft back ed.) 1455–0849 (URL: http://www.inf.vtt.fi/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	