The SMPI model: A stepwise process model to facilitate software measurement process improvement along the measurement paradigms

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieuer (Dr.-Ing.)

angenommen durch die Fakultät für Informatik
der Otto-von-Guericke-Universität Magdeburg

von: Dipl.-Inform. René Braungarten
geb. am 08.02.1980 in Stendal, Deutschland

Gutachter:
Prof. Dr.-Ing. habil. Reiner Dumke
Prof. Dr. Juan José Cuadrado-Gallego
Prof. Dr. Alain Abran

Magdeburg, den 18. Mai 2007

To Claudia, the wind beneath my wings. . .

and

To Angelika, Gunter, Marko, and Dina.

Acknowledgments

"If I have seen further than most men,
it is because I stood on the shoulders of giants."

– Isaac Newton –

Several people considerably contributed to the successful completion of this research project and the resulting PhD thesis by continuously providing me with their guidance, encouragement, and support during the years 2004–2007. I would like to express my sincerest gratitude to all of you!

First of all, I am grateful to Prof. Dr. Reiner R. Dumke, the supervisor of this research at the Otto-von-Guericke-University Magdeburg. Working on and writing this dissertation has never been a bed of roses — and without his guidance on all concerns of the research, our numerous fruitful discussions over the years, and the valuable comments he has given on the draft versions of this thesis, by far the result would not have been that successful. Moreover, Prof. Dr. Juan-José Cuadrado-Gallego and Prof. Dr. Alain Abran deserve my sincere thanks for their efforts in reviewing and providing their expert opinions on the thesis at hand.

In retrospect on the entire period that I have been working on that research project, I cannot thank enough Martin Kunz and Daniel Reitz. As PhD candidates in the Software Engineering Group we have been sharing the same office space, goals, and sorrows for years. Together we look back on interesting academic and private discussions as well as on joint research work and conference travels. All of this has resulted in a deep friendship and atmosphere of trust and mutual encouragement, which was essential in completing my thesis. Especially Martin was always pushing me a bit further towards completion. I have many good memories from these years and I hope to be able to tie up to that somewhere along the way. Good luck for your research, guys!

Furthermore, my thanks go to the other former and present team members that were instrumental in performing the work, especially Dr. Mathias Lother, Prof. Dr. Cornelius Wille, Prof. Dr. Andreas Schmietendorf, Dr. Fritz Zbrog, Dagmar Dörge, Martin Tröger, and Ayaz Farooq. I am also very much obliged to the management and employees of the host organization that offered me to validate the results of that research.

Parallel to the last year of my thesis work, I took the opportunity to work as a software quality engineer at the Bosch Rexroth Electric Drives and Controls GmbH (BRC) in Lohr am Main, Germany. It has been (and still is) very interesting to be confronted with hardware and software engineers being thirsty for practicable methodologies and research results of any kind to continuously assure the high quality of their complex products. But mastering the span between fulltime employment in industry and completing the PhD thesis meant a considerable challenge to me. Without the provident backing and flexibility of my line manager at BRC, Christian Steinisch, I would not have been able to succeed. Additionally, he was very inspirational and a great partner in coffee kitchen discussions! Thus, I want to acknowledge my deep gratitude to him.

Ultimately, let me extend my deepest gratitude to those, who made all of this possible: Starting from my earliest childhood until now, my parents, Angelika and Gunter, have attached importance to my education as a tool and prerequisite for my future live. With all their love, benignity, and belief in my capabilities they got never tired of encouraging me and offering me their invaluable moral support during all the times. I am proud to have those parents and I will be eternally grateful. Moreover, I am indebted to my brother, Marko, who never doubted me and provided vital support in hard times.

During the time of this research project the heaviest burden was put on my fiancee, Claudia. In consequence of my research work, she had do without me longer than being able to absorb weekday by weekday and weekend by weekend, especially during the last year. Thus, I highly impute Claudia that she shouldered much of my worries with patience and also got never tired to understand, motivate, and encourage me. I will also be eternally grateful to her. My final thanks shall go to my future parents-in-law, Silvia and Bernd, and my sister-in-law, Anita, for supporting Claudia and me that strong in this endeavor.

Retzstadt, June 2007
René Braungarten

This research has been in parts funded by a research scholarship of the German Federal State “Saxony-Anhalt”.

Abstract

Undoubtedly, measuring artifacts such as development processes, employed resources, intermediate deliverables, and the completed software product itself turns out to be a root discipline for the field of software engineering. Due to its importance, there are supportive documents, guidelines, and experiences with software measurement in industrial software engineering settings that can be a significant aid for organizations willing to shoulder its implementation. But establishing and sustaining a software measurement process in a particular industrial environment is often regarded as a difficult venture; these ventures are not infrequently reported to fail in practice.

It is the dedicated task of this research project to address the lack of maturity in implementing and sustaining software measurement processes in software engineering industry through a stepwise process improvement model along the measurement paradigms. In order to tackle that issue, the engineering research path is taken to be able to adopt advantageous features of prior, incomplete attempts to the problem.

Part I of the thesis deals with the observation and evaluation of those existing solutions using previously elicited criteria in terms of required content- and model-related properties of software measurement process improvement models. Starting from the evaluation's results, the most promising of the prior solution attempts is adopted as basis model and its shortcoming is analyzed.

Afterwards, in part II a development concept and a design rationale for a process model to overcome the shortcoming of the basis model are proposed, and the complemented SMPI (Software Process Improvement Model) is developed. The SMPI model is then presented graphically using diagrams according to the BPMN (Buisiness Process Modeling Notation) and textually using the EITVOX (Entry criteria - Inputs - Tasks - Validation - Outputs - Exit criteria) process modeling methodology.

Finally, in part III of the dissertation the external case-study validation of the developed SMPI is presented and interpreted by means of statistical test of hypotheses.

Contents

List of Tables

List of Figures

List of Acronyms

ACM	Association for Computing Machinery
ami	application of metrics in industry
ARC	Appraisal Requirements for CMMI
BPMN	Business Process Modeling Notation
BSc	Balanced Scorecard
CAF	CMM Appraisal Framework
CAR	Causal Analysis & Resolution
CASE	Computer-Aided Software Engineering
CBA	CMM-based Appraisal Method
CBA-IPI	CMM-based Appraisal Method for Internal Process Improvement
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
CMMI-MA	CMMI – Measurement & Analysis
CMU	Carnegie Mellon University
COCOMO	Constructive Cost Model
DMAIC	Define – Measure – Analyze – Improve – Control
DoD	U. S. Department of Defense
ESA	European Space Agency
ESPRIT	European Strategic Programme for Research and Development in Information Technology
ESSI	European System and Software Initiative
ETVX	Entry criteria – Tasks – Validation – Exit criteria
EITVOX	Entry criteria – Inputs – Tasks – Validation – Outputs – Exit criteria
FCM	Factor-Criteria-Metric
GQM	Goal-Question-Metric
GQ(I)M	Goal-Question-(Indicator-)Measure

HP	Hewlett-Packard Inc.
IDEAL	Initiation – Diagnosis – Establishment – Acting – Leveraging
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical & Electronics Engineers
IFPUG	International Function Point Users Group
iNTACS	International ISO/IEC 15504 Assessor Certification Scheme
IPD-CMM	Integrated Product Development CMM
IPPD	Integrated Product and Process Development
ISO	International Standardization Organization
IT	information technology
ITIL	IT infrastructure library
ITSM	IT service management
JTC	Joint Technical Committee
KPA	Key Process Area
M-CMM	Measurement-CMM
MBNQA	Malcom Baldridge National Quality Award
MDD	Method Description Document
MF	Measurement Framework
MIS	Management Information System
MQG	Metric-Question-Goal
NATO	North Atlantic Treaty Organization
NIST	U.S. National Institute of Standards and Technology
OMG	Object Management Group
P-CMM	People Capability Maturity Model
PAM	Process Assessment Model
PDCA	'Plan-Do-Check-Act'
PDL	Process Design Language
PDSA	'Plan-Do-Study-Act'
PIL	Process Implementation Language
PML	Process Modeling Language
PRM	Process Reference Model
PSL	Process Specification Language

PSM	Practical Software Measurement
PSP	Personal Software Process
QFD	Quality Function Deployment
QIP	Quality Improvement Paradigm
QMMG	Quality Management Maturity Grid
RCA	Root Cause Analysis
ROI	return on investment
SA-CMM	Software Acquisition CMM
SC	Sub-Committee
SCAMPI	Standard CMMI Appraisal Method for Process Improvement
SCE	Software Capability Evaluation
SE	Systems Engineering
SECM	Systems Engineering Capability Model
SEI	Software Engineering Institute
SLIM	Software Lifecycle Model
SMP	Software Measurement Program
SMPI	Software Measurement Process Improvement
SPA	Software Process Assessment
SPC	Statistical Process Control
SPI	Software Process Improvement
SPICE	Software Process Improvement and Capability Determination
SPR	Software Productivity Research
SPU	Software Producing Unit
SS	Supplier Sourcing
STSC	Software Technology Support Center
SW	Software Engineering
SW-CMM	Capability Maturity Model for Software
SWEBOK	Software Engineering Body of Knowledge
TQM	Total Quality Management
TSP	Team Software Process
VIM	International Vocabulary of Basic and General Terms of Metrology

Chapter 1

Introduction

"Research is formalized curiosity.
It is poking and prying with a purpose."

– Zora Neale Hurston –*

1.1 Background and motivation

In the era of progressive globalization, organizations of all industrial sectors have to face various challenges to stand up to the market: Competitors usually have to respond to customers' demands for low-priced but high quality products, along with extended service capabilities and severely shortened development life-cycles. These requirements are orthogonally and apparently irreconcilable. Although software is a complex product of human mental creativity and/or keen perception and comparison to conventional production is difficult, software development industry cannot be excluded from those problematic requirements. Even worse, it has to cope with additional intractable challenges, as well. The subject-matter guru Capers Jones [Jon01] characterizes the sad state of software production efforts today and subsumes: "In general, software is a troubled technology plagued by project failures, cost overruns, schedule overruns, and poor quality levels. Even major companies such as Microsoft have trouble meeting published commitments, or shipping trouble-free software."

However, Jones finds leading organizations among the ones he assessed, which can achieve better results than their fallen behind competitors and seem to have mastered software development to a large extent. Beyond certain minor factors such as excellent management, technical staff, or a cultivated professional work climate these leading organizations stand out and are able to reach software excellence, because they apply a quantitative approach to software development, which has been well-tested in other disciplines. Therefore, measurement of software development processes, employed resources, software intermediate deliverables and the completed product itself turns out to be an essential assistance in successfully tackling intractable customer requirements, in particular, and thus software production, in general. Without the means of measurement as foundation for the deduction of reliable, quantitative information those remits would fizzle, without doubt, let alone that optimizations were not possible, a fortiori. [Lig02]

*American folklorist and writer, *1903 – †1960

1.2 Research setting

This section is intended to briefly introduce the discipline of interest, highlight the main problem to be addressed by the research project as well as related sub-questions. Moreover, an abridgement of methodologies available for software engineering research is provided and mapped on the approach taken for this thesis.

1.2.1 Software engineering

As late as in 1968 the term 'software engineering' was given to a software production status workshop convened by the North Atlantic Treaty Organization (NATO) as sign of aspiration and has been used for the profession, since. The discipline of *software engineering* draws upon the knowledge of certain theories: In a retrospective of Boehm [Boe06] entitled 'A View of 20th and 21st Century of Software Engineering' a definition of the topic is given as "the application of science and mathematics by which the properties of software are made useful to people." Different from possible guesses, the sciences include not only computer science but also behavioral sciences, management sciences, and economics. While software engineering — as the term implies — mainly focuses on the development of software, computer science theory is also positioned to lay the foundation for developing, using, and improving computer hardware and software in a scientifically sound and an engineer's controlled way with the other sciences acting as foundation. That intuitive explanation is backed up by the following definition that originates from the Institute of Electrical & Electronics Engineers (IEEE) Computer Society's standard 610.12 [IEE90]:

> (1) [Software Engineering is:] *"The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.*
>
> (2) *The study of approaches in (1).*"

Being grown in the course of the last decades, that definition can be found with different flavors in the technical literature [Bau72], [Boe84], [FP97], [Pre97], [Pfl01], [Dum03], [Mun03]. In the IEEE Computer Society's Software Engineering Body of Knowledge (SWEBOK) [AMBD04] the covered areas are listed and also described as software requirements, design, construction, testing, maintenance, configuration management, engineering management, processes, tools and methods as well as software quality.

Dumke et al. [Dum05, p. 3] postulate that the entities involved in software engineering comprise methods, support tools of Computer-Aided Software Engineering (CASE), a system of measures, standards, the produced systems, gained experiences that are shared through communities. They formally describe software engineering (SE) as a system that comprises a set of elements (M_{SE}) and a set of all relationships (R_{SE}) between the elements of M_{SE}.

$$\begin{aligned} \mathbf{SE} &= (M_{SE}, R_{SE}) \\ &= (\{SE-Methods, CASE, SE-SystemOfMeasures, SE-Standards, \\ &\quad SE-SoftwareSystems, SE-Experience, SE-Communities\}, R_{SE}) \end{aligned} \tag{1.1}$$

1.2.2 Research problem

By no means, the concept of applying measurement as a root technology for the purposes of transparency and evaluation of multiple aspects in industrial software engineering environments is a new approach. It is taken for granted that it "... has one of the best returns on investment of any known software technology" [Jon01] and, additionally, reams of references report on positive aspects of so-called software measurement programs (SMPs) for organizations and characterize them for instance as [AMBD04] "... cornerstones of organizational maturity." So, software development organizations usually should not be averse to it. As with any complex task, organizations willing to shoulder the implementation and sustainment efforts for SMPs strongly depend on guidelines and process support. Accordingly, authors of the technical literature frequently suggest conducive or disavow detrimental implementation activities. Apart from industry surveys [Rub90] [Het90] [Des94] [DB99] that respectively conclude information sanitized from traceable facts which would permit inferences on conducting companies, those SMPs that come to grief are often kept secret by conducting companies for understandable reasons. The real situation with SMP is reported to be as following: According to Kaner and Bond [KB04] just a few companies establish SMPs and even fewer can finally succeed with them. Although experiences with software measurement programs are available, establishing and sustaining software measurement processes is still regarded as a difficult undertaking. [BDR96] [HG98] Corroborating, Abran et al. [ALB99] summarize the complicacy to the point and state "... software measurement programs exhibit some of the undesirable characteristics of software development projects in the sense that they are very risky undertakings in themselves. ... one could argue that software measurement programs constitute an emerging technology which has not yet reached maturity."

It is therefore the dedicated task of this research project to address the lack of maturity [BWD+04] in implementing (and thus sustaining) software measurement processes in software engineering industry through a stepwise process improvement model along software measurement paradigms applicable to any form and dimension of organization. While there have been prior attempts to that problem, to date there has been no process improvement model reported in the literature that offers step-wise improvement of the software measurement processes on the lines of models like the Capability Maturity Model for Software (SW-CMM) and/or Capability Maturity Model Integration (CMMI) of Software Engineering Institute (SEI), or International Standardization Organization (ISO)/International Electrotechnical Commission (IEC) Standard 15504 Software Process Improvement and Capability Determination (SPICE).

1.2.3 Research questions

The main research objective that is investigated by the thesis at hand is:

> *How can the implementation and sustainment of the software measurement process in industrial settings be improved?*

To address the primary research objective, several problem complexes need to be investigated. The accordingly derived research questions can thus be characterized as:

RQ1. Which are general characteristics, specific cornerstones, and best practices that form content-related criteria for a potential, stepwise software measurement process improvement model?

RQ2. Which are the requirements for process improvement models in software engineering that form model-related criteria for a potential, stepwise software measurement process improvement model?

RQ3. Find and evaluate current process improvement models that deal with the implementation of the software measurement process using criteria from RQ1/RQ2! Expose a basis model, that satisfies at least a large share of the criteria, together with its specific shortcomings!

RQ4. What would be the concept to overcome the specific shortcomings of the process improvement model finally extracted in RQ3 with respect to the criteria of RQ1/RQ2?

RQ5. How can the approach proposed as result of RQ4 be transferred into a stepwise software measurement process improvement model?

RQ6. What is the result of a case study validation in industry of the stepwise software measurement process improvement model resulting from RQ5?

1.2.4 A retrospect on research in software engineering

Unlike conventional sciences or engineering fields, in which research paradigms like double-blind studies in medicine are well-established, researchers in software engineering cannot fall back on guidance paving the way for meaningful scientific investigation and its presentation. [Sha03] Added to it, the history of computer science education was eventful in Germany. [BKD06] The short historical view on the evolution of research methods and models in software engineering provided beneath highlights the steps of evolution and the current state of guidance available for structuring and conducting this research project, in particular.

From software crisis to research crisis
In the 1950s early research on the field of software engineering was made on an ad hoc base with only a few published findings of principles that were already in place in practice. Indeed, at that time there was not yet established a separation of the research tracks of computer science and software engineering. [GVR02] As a matter of fact, starting in the 1960s the trailblazing changes of the second half of the last century, known as 'computing era', began to stumble across the problems of early software development projects in the 1970s, which were constantly over budget, exceeded their schedule, and were difficult to be controlled. [Gla94] The *software crisis* was then born. [Roy91] It culminated between the 1980s and 1990s giving rise to practitioners listening to and following whatever new theory was invented by hard-pushed researchers, who took refuge in a *advocacy research* style of frequently developing, publishing, and advocating new theories without evaluation of applicability in practice. Even worse, the typical researcher of that time did never work in industry and was stuck in that form of narrow research. Consequently, the quantity of published research results was increasing, but the quality and credibility was questionable, if not lost. When research and practice diverged from each other, analyzes indicated that the crisis shifted from practice to a *research crisis* and the mistake was slowly and painfully recognized. Different researchers advanced their opinions and spoke out hostilities expressing for instance that the discipline was not fulfilling the criteria for being a true engineering science [Sha90], that research in software engineering was 'unscientific' [FPG94], and that the whole discipline navigates between craft and science. [FG96] [Ebe97] In order to provide a way out, Tichy et al. [THP93] proclaimed that it was time for "a paradigm shift . . . from purely theoretical and building-oriented to experimental . . . " research in software engineering.

Recommended research paradigms
At the 1992 'Dagstuhl Workshop on Future Directions in Software Engineering' Adrion [Adr93] presented material on the possible *scientific*, *engineering*, *empirical*, and *analytic* research paradigms in software engineering as illustrated in figure 1.1. With a slightly different focus, Basili [Bas93] integrated scientific, engineering, and empirical paradigms to the group of experimental research, but retained the analytical research paradigm. Later in 1993, Potts [Pot93] confronted the disadvantages of the old, ill-omened research procedure he called *research-then-transfer* with the advantages of the new, promising procedure called *industry-as-laboratory*. The former procedure has its deficiencies in the limited, academic field of origin for problems to be solved, the purely technical refinement of the solution found, problems in evaluating detached researches in industrial settings due to possibly not commonly used technologies, large transfer delays, or lost confidence in academic solutions. The advantages of the latter procedure lay in the better understanding of the problem to be solved due to direct industry involvement in all research activities. Hence, a separate technology transfer phase can be omitted while early problem-focused evaluation of the proposed solution can be performed along the way. Additionally, the need for a systematic investigation of organizational social and cognitive factors for rather people-oriented than technical solutions was recognized. The premise that research ideas have to meet the test of practical application via validation [BSH86] [Gla94] [FPG94] [TLPH95] [Tic98], went into the mind of every researcher only cumbersomely. In 1995, Jeffery [Jef95] provided empirical evidence from the analysis of two software engineering journals still indicating a lack of validation activities. A subsequent study of Zelkowitz et al. [ZW98] does not draw significantly different conclusions and summarizes that besides the fact that authors often omit to characterize their research goals at the outset of the project, "validation was generally insufficient". They furthermore complain about the loose usage of terms of empirical software engineering not distinguishing between case studies, controlled experiments, and lessons learned.

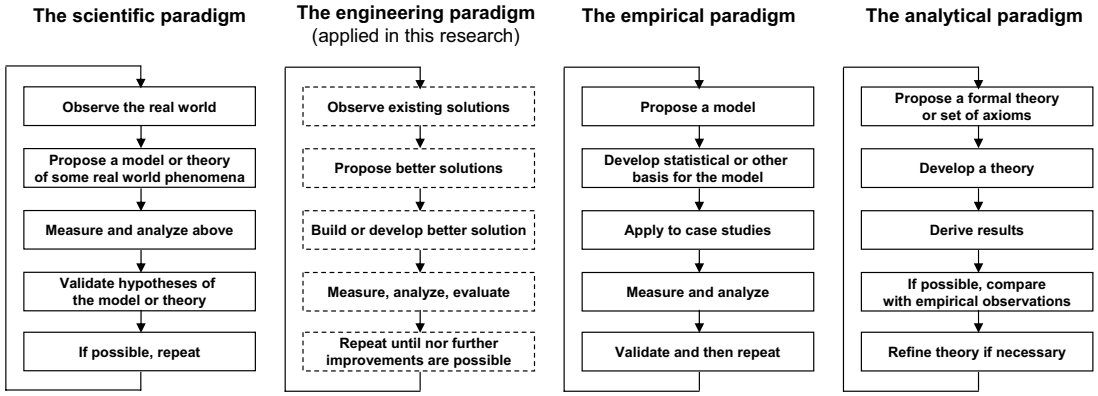

Figure 1.1: General paradigms in software engineering research (based on [Adr93])

Suggested research phases
In a subsequent paper, Glass [Gla95] seized the different paradigms again and proposed to breakdown research in software engineering into four general intervals, that is, the informational, propositional, analytical, and evaluation phase.

- **The informational phase** (Gather or aggregate information via reflection, literature survey, people/organizational survey, or poll)
- **The propositional phase** (Propose and/or build a hypothesis, method or algorithm, model, theory, or solution)

- **The analytical phase** (Analyze and explore a proposal, leading to a demonstration/or formulation of a principle or theory)
- **The evaluation phase** (Evaluate a proposal or analytic finding by means of experimentation (controlled) or observation (uncontrolled, such as a case study or protocol analysis), perhaps leading to a substantial model, principle, or theory)

Research classification patterns
Classifying a research project is made for two main reasons, Kitchenham et al. [KPP+02] dwell on: First, a certain degree of reliance of the research that readers can have in the results should be established. And second, for the purpose of meta-analyses the research should be previously categorized to be a suitable input.

Therefore, as early as in 1990, Shaw [Sha90] provided classification patterns in her preferred field of action, that is, software architecture research. Being especially suited for experimental research, Fukuda et al. [FO97] provide a detailed taxonomy. Also Zelkowitz et al. [ZW98] proposed and applied another classification scheme for methods of software engineering research as part of their literature study. Finally, presented sequentially in several papers, Glass et al. [GVR02] [GRV04] developed and applied classification patterns for computer science, software engineering, and information system research. They provide classifications for research topics, research approaches and methods as well as reference disciplines and levels of analysis, which are listed in tables 1.1 and 1.2.

Trying to induce a new debate on the connection between science and engineering in software engineering, Lazaro et al. [LM05] refer to Vincenti [Vin93] who reports on fundamental differences in the respective research problems and thus in the distinct approaches and methods applied. They put on record that "... whereas engineering problems deal with the creation of new artifacts, scientific problems deal with the study of existing ones." Being well aware of the constantly growing elicitation of research guidelines and the work for example of Shaw [Sha90] [Sha01] [Sha02] [Sha03] Wohlin et al. [WRH+00] that is being done on that field, they still recognize the immaturity of holistic research guidance in software engineering. Lazaro et al. further found out that scientific problems being either of empirical or social/cultural nature can be broached by conventional quantitative and/or qualitative research methods. Because of intercessors like Basili [Bas96] that have promoted the use of the scientific paradigm calling for more experimentation and validation, sophisticated guidelines for doing empirical research have been developed for instance by Kitchenham and colleagues [KPP+02] who draw parallels to research in medicine.

But for broaching engineering problems no precise methodology exists, so far, because the major engineering component 'human creativity' complicates universal approaches. Fortunately, for the issue of research of engineering problems, Klein et al. [KH03] propose a promising method. It is similar to the engineering research procedure proposed by Adrion [Adr93]: Starting with the study of existing solutions, their advantages and shortcomings are recognized and used to design a better proposition thereby retaining all the advantages but dismissing as much disadvantages as possible.

Topic categories			
1	Problem-solving concepts	6	System/software management concepts
1.1	Algorithms	6.1	Project/product management (incl. risk management)
1.2	Mathematics / computational science	6.2	Process management
1.3	Methodologies (object, function/process, ...)	6.3	Measurement/metrics (development and use)
1.4	Artificial intelligence	6.4	Personnel issues
2	Computer concepts	6.5	Acquisition of (packaged/custom) software
2.1	Computer/hardware principles or architecture	7	Organizational concepts
2.2	Intercomputer communication (networks, ...)	7.1	Organizational structure
2.3	Operating systems	7.2	Strategy
2.4	Machine/assembler-level data/instructions	7.3	Alignment
3	Systems/software concepts	7.4	Organizational learning/ knowledge management
3.1	System architecture/engineering	7.5	Technology transfer
3.2	Software life cycle/engineering	7.6	Change management
3.3	Programming languages	7.7	Information technology implementation
3.4	Methods/techniques (patterns, process models)	7.8	Information technology usage
3.5	Tools	7.9	Management of “computing” function
3.6	Product quality	7.10	IT impact
3.7	Human-computer interaction	7.11	Computing/information as business
3.8	System security	7.12	Cultural/legal/ethical/political implications
4	Data/information concepts	8	Societal concepts
4.1	Data/file structures	8.1	Cultural implications
4.2	Data base/warehouse/mart organization	8.2	Legal implications
4.3	Information retrieval	8.3	Ethical implications
4.4	Data analysis	8.4	Political
4.5	Data security	9	Disciplinary issues
5	Problem domain-specific concepts	9.1	“Computing” research
5.1	Scientific engineering (incl. bioinformatics)	9.2	“Computing” curriculum/teaching
5.2	Information systems (decision support, ...)		
5.3	Systems programming		
5.4	Real-time (incl. robotics)		
5.5	Edutainment (incl. graphics)		

Table 1.1: Classification of research topics in computer science, software engineering, and information systems (adapted from [GVR02, p. 495])

Research approaches	Research methods	Reference disciplines	Level of analysis
Descriptive - Descriptive system - Review of literature - Descriptive other Evaluative - Evaluative-deductive - Evaluative-interpretive - Evaluative-critical - Evaluative-other Formulative - Formulative-concept - Formulative-framework - Formulative-guidelines/standards - Formulative-model - Formulative-process, method, algorithm - Formulative-classification/taxonomy	- Action research - Conceptual analysis - Conceptual analysis mathematics - Concept implementation - Case study - Data analysis - Ethnography - Field experiment - Field study - Grounded theory - Hermeneutics - Instrument development - Laboratory experiment – human subjects - Literature review/ analysis - Laboratory experiment – software - Mathematical proof - Protocol analysis - Simulation - Descriptive/ exploratory survey	- Cognitive psychology - Social and behavioral science - Science - Economics - Management - Management science - Mathematics - Other - Not applicable - Self reference	- Society - Profession - External business context - Organizational context - Project - Group/team - Individual - Abstract concept - System - Computing element

Table 1.2: Classification of research approaches, methods, disciplines, and levels of analysis (adapted from [GRV04, p. 91])

1.2.5 Validation methods in empirical software engineering

Undoubtedly, an empirical validation of software engineering research results is desirable to obtain confidence in the proposed theoretical structures. Accordingly, a variety of different special-purpose methods for conducting empirical studies has been proposed and a certain experimental terminology could be established as it is e. g. recapitulated by Kitchenham et al. [KPP95]. But to be able to choose the most appropriate empirical validation approach, the purpose of the study, its group of participants, intended observations, and the way of analyzing the observations have to be considered. [Ari01]

The probably earliest proponent of experimental, that is, empirical research in software engineering, Basili [Bas93], describes an approach to experimenting characterized by the scope in terms of the number of teams used for the replication of the project and the number of distinct projects under analysis; the related four experimental treatments (cf. table 1.3) are *blocked subject-project*, *replicated project*, *multi-project*

variation, and *single project case study*. With respect to different notes of Kitchenham et al. [KPP95] [KPP+02], today the most renowned and thus established validation methods in empirical software are, in descending order of their scopes, *surveys*, *experiments*, and *case studies*. To select an adequate validation approach out of this set, a brief reflection on it shall ease making a choice.

	One project	**Multiple projects**
One team per project	Single project (case study)	Multi-project variation
Multiple teams per project	Replicated project	Blocked subject-project

Table 1.3: Scopes of empirical evaluation (adapted from [Bas93])

Surveys — research in the large
Survey research in its intrinsic form is scientifically advantageous to gather large scale information from a sample of a population of a significant number of projects, thereby applying standardized instruments or protocols. [Jar04] Because of the multitude of projects examined, this kind of evaluation is called *research in the large* [KPP95]. Kraemer [Kra91] characterizes doing survey research as following:

1. Some aspects of interest of a study's population are examined with the aid of a survey that has been designed to gain quantitative observations.

2. In general, the data collection is performed by the principal means of gathering answers to structured, predefined questions.

3. The data collection is solely directed at a portion of the study's population (sample) and the circumstances facilitate the generalization of the findings to the entire population.

Imposed by the targeted quantity of observations, the potential benefit of a survey research's representativeness in terms of generalizing the findings, that is confirming or falsifying a proposed theoretical structure, to many projects and/or organizations stands to reason. While most often a questionnaire is the means of choice for survey research, it is just one data collection technique among others, which can demonstrate association but fails in establishing a relationship to causality. [Kit96c] Another problem may arise when the premise of random samples is violated by a researcher, who might rather prefer samples being sympathetic to the research object under study *convenience samples* than real *random samples* possibly leading to a refutation of hypothesis. [Cun97] After all, the consistency of a survey's questions formulation is an important factor contributing to the validity of the entire research. [Ari01]

Guidelines and/or proven criteria for consistently formulating a questionnaire are given by Kerlinger et al. [KL99]

Experiments — research in the small
In order to control as many factors as possible, which may have an impact on the research object of interest, formal and carefully controlled experiments are applied. Because the necessary control of these factors is most probably cost-intensive the experiments can often only take place in small scale studies and are hence called *research in the small*. [KPP95] For a certain amount of time, this approach has been the preferred way for confirming or falsifying hypothesis and/or theories. [Ari01] Unfortunately, besides problems of assembling an appropriate and pristine sample the mapping of findings from controlled experiments on an environment outside the origin is at least questionable, if not impossible at all. [KPP95] Because conducting formal experiments

was very widespread and has proven to be successful several times [BC91] [BMA96] [ALB99] [Apr05], its proponents like Basili et al. [Bas93] [BSH86] or Pfleeger [Pfl95a] provide a good deal of support on how to conduct them.

Case studies — research in the typical
Observation: Case study or *observation* research as defined by Yin [Yin02, p. 13] deals with "an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between phenomenon and context are not clearly evident." Owing to the fact that this is the bread-and-butter research method to gain a deep understanding of a certain case being part of e. g. a development project, Kitchenham et al. [KPP95] denominate it *research in the typical*. Moreover, case study research using qualitative and quantitative techniques is of elevated importance for the evaluation of software engineering methods and tools in industrial settings, because scale-up issues can be circumvented. [KPP95] [Ari01] Despite all the good, there is also a negative side to be mentioned: Not only that the scientific rigor familiar from surveys and formal experiments cannot be achieved by case studies [KPP95], more or less the generalization of findings turns out to be questionable due to either the lack of control or sudden influences. Potts [Pot93] complains: "When working on a case study, there is always a nagging doubt that it is not representative."

However, conducting a case study can be very significant for validation aspects and provided that guidelines for instance of Kitchenham et al. [KPP95] or of Arisholm et al. [AAJS99] exist, its success seems to be very likely.

Action research: In contrast to case study research, where almost no intervention from the researcher is needed, Argyris et al. [AS91] classify empirical studies in which the pertinent researcher is about to bring deliberate changes to the affected organization as *action research*. Avison et al. [ALMN99] expand this intuitive understanding of this approach when they notice: "Action research combines theory and practice (and researchers and practitioners) through change and reflection in an immediate problematic situation within a mutually acceptable ethical framework." Baskerville et al. [BWH96] characterize doing action research as following:

1. Resulting in the anticipated benefit for the researcher and the organization under study, the researcher is actively involved in the study on site.

2. Because action research is based on the idea of an actively participating researcher, who recognizes and refines new cognitions with respect to an conceptual framework, immediate usage of obtained knowledge is aimed at.

3. Forming a link between theory and practice, the research is conducted in cycles.

Regrettably, a major challenge of this approach, which was not at all used in software engineering at least until the report of Glass et al. in 2004 [GRV04], exists. There are detractors, who are emphatic on their prejudice of the researcher's lack of impartiality [BWH96] and complain about "consulting masquerading as research". [Gum99]

But with the addition of an explicit statement of the approach taken, the goal of the research, the proposed theory, and method at the very beginning of the research and in contributing publications, these critiques can be disposed of, once and for all. [Rob97] Guidelines and a five-step cycle related to the action research process are provided by Susman et al. [SE78]

1.2.6 Classification of the research project

Recognizing the importance of further classifying the software engineering research according to the topic, utilized approaches, referenced disciplines, and the targeted level of analysis as argued for by Glass et al. [GVR02] [GRV04], these facts shall not be left out and are stated as following:

- **Research discipline:** Software engineering
- **Research topics:**
  - 6.1 Process management
  - 6.2 Measurement/metrics
  - 7.7 Information technology implementation
  - 7.8 Information technology usage
- **Research approaches:**
  - Descriptive: Review of literature
  - Formulative: Formulative-model
- **Research methods:**
  - Literature review/analysis
  - Case study
- **Referenced disciplines:**
  - Mathematics
  - Social and behavioral sciences
- **Level of analysis:** Organizational context

1.2.7 Striking the engineering research path

While several sketchy solutions to the problem of finding a panacea already have been proposed and seem to exist for years, they could obviously not help in paving the way for successfully implementing and sustaining SMPs in industrial software engineering settings. With respect to Klein et al. [KH03] who suggest a similar methodology for solving engineering research issues, the engineering research paradigm provides an adequate, viable framework for the development, evaluation, and further improvement of a proposition including all advantages but excluding all shortcomings of prior solutions. Seizing the critical suggestions of the software engineering pioneer and luminary, Watts S. Humphrey [Hum02, p. 50], the approach of improving a suboptimal solution to the research problem is favored for this academic project:

> "Every time software people have faced a new problem, instead of building on prior work, we have invented some new language, tool, or method. This forces us to start over, and it also forces our users to start over."

By and large, the research project followed the given structure of the engineering research paradigm. [Adr93] [Bas93] The research cycle, as its synopsis is described beneath, was run through once:

I Observation of existing solutions
The research step of observing existing solutions was threefold: First important aspects of software measurement programs in industrial settings had to be extracted to propose content-related evaluation criteria for existing solutions. Second, based on the fundamentals of software process engineering and a review of characteristics of mainstream Software Process Assessment (SPA)/Software Process Improvement (SPI) models in software engineering as well as figures of merit set by the up-to-date ISO/IEC Standard 15504, model-related evaluation criteria for existing had to be formed. Third and finally, the related implicit or explicit work and its respective appropriateness for the improvement of software measurement process implementation and sustainment called for an assessment against these criteria and for choosing a promising model to adopt.

II Proposal and development of a better solution
As the result of the shortcoming of a selected existing solution as extracted by the steps before, a concept for improving the selected basis solution was to be formulated and afterwards imbued with life. Apart from consensus mapping of reference literature, several different techniques like informal subject-matter expert interviews, brainstorming sessions with chummy researchers, or industry and conference presentations with subsequent, fruitful discussions were conducted to provide the model under development. The gained experiences and critiques were incorporated and lead to the improvement of the model as is.

III Measure, analyze, evaluate
Having the confidence to be right from the theoretical point of view because of the voluminous literature study, the model had to demonstrate its usefulness in a case study in an industrial environment as practical validation background.

1.3 Structure of the thesis

While the introductory chapter of the thesis at hand dealt with general aspects such as the addressed problem's motivation and the research methodology, the subsequent chapters will focus on the research and its validation. The remainder of the thesis is structured in the following way:

Chapter 2 examines and investigates important aspects of SMPs in industrial settings and proposes content-related, mandatory evaluation criteria for existing solutions of software measurement process improvement models.

Chapter 3 examines, based on the fundamentals of software process engineering and a supplementary review of characteristics of mainstream SPA/SPI models in software engineering as well as figures of merit set by the up-to-date ISO/IEC Standard 15504, model-related, mandatory evaluation criteria for existing solutions of software measurement process improvement models.

Chapter 4 briefly reviews the related implicit or explicit solutions of software measurement process improvement models and evaluates its respective appropriateness against the content-related and model-related specific criteria as revealed in both chapters before. Furthermore, the most promising model of the evaluation is selected as basis and its shortcomings to be addressed by this research are exposed.

Chapter 5 seizes the shortcoming of the basis model as selected in chapter four and proposes a concept to overcome it. Moreover, the design and development rationale for this concept is conceived and the development of the SMPI model being complementary to the selected basis model is presented.

Chapter 6 reflects the external validation endeavors on the base of a case study in an industrial setting. Furthermore, it discusses the experiences gained during the performance of the validation.

Chapter 7 gives a summary of the research's contributions compared with the questions asked, and outlines prospects.

Part I

Observation of existing solutions

Chapter 2

Measurement in software engineering industry

"I don't see how we can have software engineering in the 21st century without measurement."

– David N. Card [McC00] –*

2.1 Introduction

Due to measurement being apparently one key to success and sophistication of any engineering discipline, the area of software engineering must not be and is not an exception in trying to apply measurement to this end, actually. [KBS94] [PJCK97]

As many endeavors to translate the concepts of classic measurement theory to the area of engineered software production, operation, and maintenance have been undertaken during the last two decades like the one of Wang [Wan03], they have also become documented in the professional literature. But, software measurement, being a relatively young discipline within software engineering, still suffers from some teething troubles as any other young discipline in engineering or science does. [BMB02] García et al. [GBC+06] corroborate this assertion and mention that "...software measurement is currently in the phase in which terminology, principles, and methods are still being defined, consolidated, and agreed." Furthermore, the literature review on measurement in software engineering gives also good evidence of the discipline's shaky terminology, methods, and concepts. In numerous references important researchers of that area tried to coin different terms for the application of measurement to software engineering, from which some have been proven obsolete by their own eponyms. Those denominations are e. g.: *software metrication* [FW86], *software engineering measurement* [BMB96] [FP97] [Mun03], *software measurement* [FW95] [OJ97] [AD99] [DA99] [Cla02] [AMBD04] [LB06], *computing measurement* [KB04], *software metrics* [CL95] [HS96] [FN99] [AM06].

Based on the fundamentals of measurement theory as recognized by the author and presented accordingly in appendix A, this section will at first clarify the terminology of measurement in software engineering as far as it will be used in this thesis and then present assorted concepts and principles.

*Technical lead for development of Practical Software Measurement, editor-in-chief of the Journal of Systems and Software, listed as top U. S. researcher in the *Who's Who in Science and Technology*

2.2 Clarification of terminology

In textbooks, articles, and papers on metrology in traditional sciences like physics or mathematics as well as in software engineering some authors use the terms 'measurement', 'measure', and even 'metric' interchangeably as for example observed by Jacquet and Abran [JA97]. They also revealed that the noun 'measurement' is not only being interpreted from a procedure's name perspective, but as the application of that procedure, its result, or the design and exploitation of the method.

In 2002, in their papers Abran and Sellami examined [AS02a] [AS02b] the body of knowledge in metrology as manifested in the 1993 version of the International Vocabulary of Basic and General Terms of Metrology (VIM) of ISO/IEC. [ISO93] In doing so, they ascertained that it represents an "official national and international consensus", but complained about the absence of its consideration in their professional community, which is concerned with measurement research and application in development, operation, and maintenance of software. To pave the way for recognition and understanding of the vocabulary's textually described number of 120 terms and inter-related concepts in their community, Abran and Sellami presented one high-level and several low-level models in a graphical representation.

Following up these efforts, Martin et al. [dlAMO03] and afterwards García et al. [GRB+04, GBC+06], who received suggestions by Professor Abran, presented a proposal to address the need for a common terminology of all audiences involved in software measurement, that is, practitioners, researchers and standard developers. Therefore, they examined software engineering international standards provided by major standardization bodies, such as IEEE Standard 610.12 [IEE92], ISO/IEC VIM [ISO93], IEEE Standard 1061 [IEE98], ISO/IEC 14598 [ISO01], ISO/IEC 9126 [ISO04], ISO/IEC 15939 [ISO02], Practical Software Measurement (PSM) [MCJ+01], and others. The authors compared the pertinent definitions and produced a preliminary ontology of the terms concerned with measurement in software engineering in order to provide a basis for discussion among researchers and practitioners. Once, that ontology will have been completed and fully agreed, it could serve as a foundation for harmonization efforts of international standards in terms of metrology.

Over and above, other parts of the recognized literature for this thesis also contribute to the understanding of the denominations and will be consulted to clarify the terminology. Hence, in order to base that research on definite wording, the terms 'software measurement', 'software measure', 'software metric', and 'software meter' are clarified based on the results of the literature study. For any other terms the reader should refer to Garcia et al. [GBC+06]

Software measurement
There are numerous, frequently used definitions in the literature around the term *software measurement* or similar neologisms. [Rag95] [FP97] [IEE98] [Pre97] [Mun03] [KB04] However, a unified meaning of the compound noun, that corresponds to the appendix, section A.4.3 is to be established, thereby expelling the separate occurrence of the term 'measurement' in the context of software engineering.

> **Definition 2.1:** *Software measurement* is the procedure of an empirical, objective assignment of numbers or symbols, according to a rule derived from a model or theory, to attributes of instanced software engineering entities with the intent of describing them.

The software measure vs. To measure software
Again, there is an ambivalence in the meanings of the term's constituent 'measure', since it can also be used as both, a verb and a noun. While the verb characterizes the application of software measurement, the noun describes its result. [BBFG90] [Fen91] [Het93] That ambiguity is to be compensated by expelling the singular usage of the word 'measure' and shall be clarified with the aid of the following two definitions that correspond to the appendix, section A.4.3:

> **Definition 2.2:** The verb *to measure software* denominates the activities of software measurement.

> **Definition 2.3:** The noun *the software measure* describes the result of the execution of activities of software measurement.

Software metric
Primarily originating from the ancient Hellenic noun 'métron' (measure), the term 'metric' denominated since the time of the old Hellenic poets like Homer or Hesiod the art of measurement of verses in poetry, actually known as the theory of tact and tact stressing. [Raa86] Nowadays, the term is widely used and interpreted in areas different from poetry such as music, chemistry, mathematics, and software engineering, of course. One might probably suspect that the meanings in mathematics and software engineering match, but that is actually not the case.

In mathematics, more specifically in the subarea of analysis and/or geometry [Die68], a *metric room* (M, d) is defined as a set M provided with a function $d : M \times M \rightarrow [0, \infty)$ on it satisfying the following axioms for all elements x, y, z of M:

- $d(x, y) \geq 0, d(x, y) = 0 \Leftrightarrow x = y$ (definite)
- $d(x, y) = d(y, x)$ (symmetric)
- $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

A function d specifies the distance between in each case two points of a plane, x and y, and is thus in analysis called a *metric* on M. Those distance metrics are frequently used in mathematics in both, Euclidean (i. e. Euclidean distance) and Non-Euclidean geometry (i. e. Manhattan and/or Minkowski metric). [Kra87]

Though, software measurement is a procedure that, when following the principles of general measurement theory as presented in appendix A, a software measure results from the predefined mapping process between a numerical and an empirical relational system within software engineering. The concept underlying software measurement completely differs from the one of a distance function such as the Hamming distance [Ham50] in information theory between strings of equal length. From the theoretical point of view, these differences must not be mixed up by using the terms 'software metric' and 'software measurement' interchangeably. However, the term is widely established under North American practitioners like Grady [GC87], Hetzel [Het93], or Goodman [Goo04] and internationally recognized researchers like Kitchenham [KL87], Pigoski [Pig97] or Kaner [KB04]. For instance, Kaner asserts that a "metric is a measurement function". Many authors of textbooks or articles on software measurement like Conte [CDS86], Baker [BBGM87] [BBFG90], Melton [MGBB90], Offen and Jeffery [OJ97], or Fenton and Pfleeger [FP97] advise for cautious and sensible exposure with the term. Fenton and Whitty [FW95, p. 6] back up that opinion and state that "... much work was criticized for its failure to adhere to the basic principles of measurement that are central to the physical and social sciences." Corroborating that indictment, among others, e. g. Whitmire [Whi97] or Zuse [Zus98] strictly, and in fact

correctly, decline the usage of the term 'metric' because being reserved for purposes of geometry in mathematics. Hereunto Whitmire argues on page 208: "The use of the term 'metric' for any other type of measure is imprecise at best and misleading at worst." However, it is felt that the definition of IEEE's Standard 610.12 [IEE90] Glossary of Software Engineering Technology does contribute to irritation because it defines a metric as "a quantitative measure of of the degree to which a system, component, or process possesses a given attribute." So, even though the term 'metric' is imprecise or misleading from the measurement theory point of view, as can be seen, it has encroached upon (especially) North American professionals and their standardization communities like the IEEE or the Association for Computing Machinery (ACM), which are concerned with aspects of software engineering measurement. But García et al. [GBC+06] report on current harmonization efforts on those standardization bodies, that will "avoid the use of metric in order to be aligned with the rest of measurement disciplines, which normally use the vocabulary defined in Metrology" (VIM).

Nevertheless, ignoring its existence and strictly deleting it from the register cannot be a satisfying solution and shall be overcome by adopting its meaning in practice. Fenton and Pfleeger [FP97, p. 14] share that view and state: "Software metric[s] is a term that embraces many activities, all of which involve some degree of software measurement." Goodman [Goo04, p. 6] provides a definition stressing the practical use of software measurement for management. It is therefore accepted with merely minor modifications:

> **Definition 2.4:** A *software metric* is the continuous application of measurement-based techniques to software engineering processes and its products or artifacts to supply meaningfully and timely management information, together with the use of those techniques to improve those processes and its products or artifacts.

Software meter

In general, a meter or indicator can be used to draw an individual's attention to a particular situation. In particular, when software engineering processes are performed in the scope of projects, project managers are urged to manage their projects in a stringent corsage of given economic, temporary, and qualitative constraints or conditions. [PM03] *Software meters* or *indicators* are the results of comparisons between those project constraints or conditions and software measures, which mirror the current state of some software engineering project entity of interest that might be eligible for a constraint. [Het93] [Car93] [Rag95] Then, these results can be interpreted and used for decision making in the project. [Pre97] IEEE's Standard 610.12 [IEE90] Glossary of Software Engineering Technology provides a disambiguation that shall be adapted with some alignment, here:

> **Definition 2.5:** A *software meter* or indicator is a device or variable, such as a flag or semaphore, that reflects the result of analysis or comparison between the current state of some software engineering project entity gained by software measurement and a specified project constraint.

2.3 Entities and attributes of interest

Referring to definition 2.1, software measurement is intended to quantify attributes of instances of entities that are either used as means or inputs or that are in some way produced by and thus also involved in software engineering processes. However, as

already experienced during the review of related literature for other aspects, no real consensus on how to classify those entities for software measurement could be found.

Classification of entities
Initially in 1991, Fenton [Fen91] presented a framework comprising the three classes of software engineering entities relevant for measurement, namely, *processes*, *products*, and *resources*. Daskalantonakis [Das92] exchanges resources with *projects*. In later publications [Fen94] [FP97] Fenton, again seized and further substantiated his original classification. Whitmire [Whi97] enhanced Fenton's framework and added the class of *projects* to it. Because the classification of Fenton is probably the most verisimilar one, it is adapted for this thesis. However, the conglomerate of the process, product, and resources entities as 'project' entity is highlighted in the last section. There is another view of Munson [Mun95, Mun03] that shall be mentioned, as well: He accepts the Fenton framework's classes 'processes' and 'products', but splits the virtual class 'resources' into 'people' and 'environment'.

Classification of attributes
When trying to classify attributes of entities, one usually falls back to the classifications of measurement of those attributes as given in the appendix, section A.5: fundamental and derived, extensive and intensive, or subjective and objective attributes.

2.3.1 Process entity

Based on the deeper insights of researchers with pioneering tasks like Watts S. Humphrey [Hum88, Hum89] the software engineering community has been painfully made believe that "product quality is evidence of process success." [PJCK97] But what is the entity 'process'? There is an often cited definition in literature of Fenton [Fen91, p. 43]: "[A process is a set of] ... software related activities which normally have a time factor." However, it seems to be inappropriate because it implies the existence of a time-period like a project as prerequisite. While projects are per se nonrecurring, processes and especially the notion of process improvement are based on (potentially revised) repeatable activities and/or processes. In order to eliminate this misreading, a sound definition from a software engineering point of view is required. Despite a description of the term can merely be found with different flavors in related publications [Hum88, Hum89] [Lon93] [KJ03] and even SWEBOK [AMBD04] recognizes the term's equivocation, a definition of Pall [Pal87, p. 27] shall put it in contextual perspective:

> **Definition 2.6:** "A process can be defined as the logical organization of people, materials, energy, equipment, and procedures into work activities designed to produce a specified end result."

Among other process constituents, in fact, there are *small-grained* procedures or activities of an area [KBD05] [XXN+06] which shall be characterized appropriately by certain attributes of interest, like e. g. the elapsed time, the expended effort, or the number of requirements changes for the requirements specification process. *Large-grained* initiatives of software process quantifications or rather assessments such as for instance the SW-CMM of SEI at the Carnegie Mellon University (CMU) [PCCW93a] integrate measures of those small-grained activities of all software engineering processes to assess an organization's processes and at the same time identify a way for improvement in areas, in which the software measures indicate weaknesses.

Dumke et al. [DSZ05, p. 8] formally summarize their understanding of a software development process (SD) as cooperation between entities from development methods, the lifecycle, or software management and required resources (SR) for the purpose of producing a product:

$$\begin{aligned} \mathbf{SD} &= (M_{SD}, R_{SD}) \\ &= (\{developmentMethods, lifecycle, softwareManagement\} \cup M_{SR}, \\ &\quad R_{SD}) \end{aligned} \tag{2.1}$$

Selected examples for different types of process attributes are listed in table 2.1.

Type of attribute (measurement)	**Instance of process entity**	**Example**
Fundamental	Detailed design	Time elapsed
Derived	Maintenance	Mean cost per introduced error
Extensive	Coding	Resource usage
Intensive	Testing	Appropriateness felt by staff
Subjective	Testing	Process stability felt by staff
Objective	Requirements	Number of elicited requirements

Table 2.1: Examples of selected process attributes

2.3.2 Product entity

Long before the quantification of process entities has come into vogue, measurement of software engineering products was encouraged by customers, which are usually greedy for information on the final product's characteristics. [PJCK97] The field of software measurement concerned with the software products or any intermediate artifacts of it is a broad one: Starting for instance relatively early in a product's life-cycle the quality of requirements specifications can be assessed. [MD03] Then, one could want to have quantitative information about the emerged design's complexity [HK81] [CA88] or several object-oriented measures of design and code artifacts [CK94] (An overview of numerous proposed software measures applicable to software design is given by Yu and Lamb [YL95]). Finally, after determining the source code's cyclomatic complexity [McC76], both, a number of quality assessments according to ISO/IEC 9126 [ISO04] [FBD05] [BFK+06] [KSBD06] [FBK+06] and the risk at the software's release time or its reliability could be measured and/or predicted [MIO87]. The above examples are only a small excerpt of the instantly growing number of attributes which can be measured of the multiple intermediate and final artifacts in software engineering.

Again, a formal definition of Dumke et al. [DSZ05, p. 4] shall be used to formally define a software product (SP). From a general point of view, the potential entities of a software product consist of any form of software programs and documentation:

$$\begin{aligned} \mathbf{SP} &= (M_{SP}, R_{SP}) \\ &= (\{programs, documentations\}, R_{SP}) \end{aligned} \tag{2.2}$$

Selected examples for different types of product attributes are listed in table 2.2.

Type of attribute (measurement)	Instance of product entity	Example
Fundamental	Code	Size (as per lines of code)
Derived	Design	Modularity
Extensive	Specification	Syntactic correctness
Intensive	Specification	Comprehensibility
Subjective	Object code	Usability
Objective	Test data	Coverage level

Table 2.2: Examples of selected product attributes

2.3.3 Resource entity

Several decades of experience [Wei71] [DeM82a] [DL99] show that especially the variations in mental conditions of *human resources* are likely to be a main reason for variations of the resulting software product's quality or complexity. In 1983, Basili and Hutchens [BH83] reported such a case from real-life.

Apart from human resources, also *non-human resources* like monetary budget, time schedules, available development tools, other production equipment and the working environment are important factors somehow contributing to the resulting software product, too. Dumke et al. [DSZ05, p. 6] provide a formal definition of the resources (SR) used in software development:

$$\begin{aligned} \mathbf{SR} &= (M_{SR}, R_{SR}) \\ &= (\{personnelResources, softwareResources, hardwareResources, \\ &\quad financialResources\}, R_{SR}) \end{aligned} \tag{2.3}$$

Software measurement in that area does not only focus on quantification but also on the prediction of those resource factors. Among the tools and techniques that are used for effort prediction, range e. g. Boehm's Constructive Cost Model (COCOMO) [Boe81] and its updated successor COCOMO 2.0 [BCH+95] and/or COCOMO II [BHM+00]. Beyond that, also Putnam's Software Lifecycle Model (SLIM) [Put78] and possible variations of Albrecht's [Alb79] [AG83] primary function points are often applied for reasons of effort estimation in software engineering.
Selected examples for different types of resource attributes are listed in table 2.3.

Type of attribute (measurement)	Instance of resource entity	Example
Fundamental	Hardware	Memory size
Derived	Software	Cost per month of usage
Extensive	Personnel	Age
Intensive	Personnel	Intelligence
Subjective	Office	Temperature
Objective	Team	Structuredness

Table 2.3: Examples of selected resource attributes

2.3.4 Projects as conglomerate of entities

As a conglomerate of all software engineering entities of interest for software measurement, the 'project' stands out. Gray [Gra81] defines projects as "a complex of non-routine activities that must be completed with a set amount of resources within a set interval." Starting from this definition, Whitmire [Whi97] describes the term by interrelating the problem to be solved by the software *product*, internal or external *goals* or *standards*, *processes*, *techniques*, *resources*, internal and external constraints as well as the software *product* itself. Because project goals and available resources can have a material impact on the result of the project, two projects working on the same problem can result in different products. This has been corroborated empirically several times in literature as for instance by the evaluation of *Weinberg's Coding Wars*. [DeM82a]

Owing to the project's multi-layer arrangement, which acts as the glue between the instances of process, resource and product entities and can be best described as the cross-section of all, software measurement for the project entity can be fairly complex: In addition to software measurement for the applied processes, expended resources and the product together with its intermediate artifacts to assess the performance related to project goals, standards, and requirements, progress and effective, commercial project performance comparisons with schedules and given budgets are required. Project level measures can assist in that [Jon01] as e. g. Pearse et al. [PFO99] report of a case at Hewlett-Packard Inc. (HP).

Based on the formal notation as proposed by Dumke et al. [DSZ05], a software development project (SDP) is defined for the context of this thesis as following:

$$\begin{aligned} \mathbf{SDP} &= (M_{SDP}, R_{SDP}) \\ &= (M_{SD} \cup M_{SP} \cup M_{SR}, R_{SDP}) \end{aligned} \tag{2.4}$$

The examples for different attributes corresponding to the project entity arise from its constituents and have been already covered. For further reading on this topic the interested reader is referred to the paper titled *A Framework for Software Project Metrics* of Woodings and Bundell. [WB01]

2.4 The importance of software measurement

In his textbook named *Measuring and managing performance in organizations*, Austin [Aus96] exposes the central role of measurement for organizations not only in software engineering but across the board, thereby emphasizing people-related aspects. Withal, Austin highlights both, the *intended functions* and the *dysfunction* of (software) measurement, in the last resort.

When studying other related literature for benefits and dysfunction of software measurement, only a partition of the general aspects introduced by Austin are examined and the rest is missed out. Thus, at first the generally intended measurement functions and also occurring negative dysfunctional behavior are presented. Then, different views and the potential consumers of measurement data are specified before revealing the state of recognition as reported in software measurement literature.

2.4.1 Intentional functions and negative effects

Motivational measurement
Organizations of any industry branch have always been interested in increasing their

employees' efficiency in terms of goods produced or services performed by them within a set period of time. In order to judge, whom to reward for performing beyond the call of duty and whom to punish for decreased efficiency, they quantify their staff's performance. Because it is manifested in the nature of men to perform better when the anticipated work results are being observed, organizations might want to affect or rather motivate their staff to provoke greater expenditure of effort in pursuit of organizational goals.

The described psychological phenomenon became documented as early as in 1939 by Roethlisberger and Dickson [RD39] as the *Hawthorne effect*. This term was coined because certain psychological studies at the Hawthorne Works of Western Electric Company in Chicago, Illinois, USA were performed to investigate the impact of environmental, social, and other factors on the staff's general behavior in a period between 1924 and 1933. As a result, the studies revealed common reactive behavioral changes of individuals, once they recognize special treatment or (unintended) participation in experimental conditions. Because this effect is associated with these trailblazing studies, it is commonly called Hawthorne effect.

Informational measurement
The most often primarily intended use of software measurement is, without doubt, its informational purpose. With its aid the necessary logistical, status, and research information can be retrieved. Austin distinguishes informational measurement further into *process refinement measurement* and *coordination measurement*.

On the one hand, *process refinement measurement* as pioneered by Taylor [Tay16] is being used in organizations to gather detailed insights into the structure of the internal workings of activities and/or processes applied. This helps in both, long-term management and organizational process improvement by quantitatively revealing, which processes are suboptimal and need to be redesigned. On the other hand, *coordination measurement* is being performed in organizations for its purely logistical purpose. It is thought to provide the information necessary for real-time management of flows and schedules of the organization. This helps senior and middle management in deciding when and where to reserve additional or cut excrescent resources.

Dysfunction of measurement
Beyond the positive and intended aspects of measurement in organizations, the tide may also turn and do psychologically and often consequently economically more harm than good. Blau [Bla63, p. 10] calls this phenomenon *dysfunction* and defines it as "those observed consequences of social patterns that change existing conditions in the direction opposite to socially valued objectives, or consequences that interfere with the attainment of valued objectives." Based on this definition, Austin [Aus96, p. 10] interprets dysfunctional behavior as "consequences of organizational actions that interfere with attainment of the spirit of stated intentions of the organization."

Blau further states that this effect can be provoked when measures are not perfectly related to the abstract concept they are intended to measure. Believing in his notes, an ill-designed measurement system failing to measure the actually important aspects is the reason for dysfunctional measurement. Owing to several years of experience, Blau recognizes that even the most sophisticated and best-designed measurement systems do not necessarily prevent individuals from reacting with dysfunctional behavior.

2.4.2 Aspired value

As can be recognized from the notes in section 2.3 above, software measurement in general deals with the quantification of attributes of process, product, and resource entities or their aggregation in project entities. In its application in commercial software production organizations, possibly elaborate and cost intensive initiatives of software measurement cannot be legitimated to senior management without having tangible objectives in mind, which actually bear chances to pay out and be profitable. That is, software measurement for the sake of bureaucracy will most probably very soon fail. Isolated measurement for motivational issues is questionable, too. But software measurement does make sense, if is introduced to cover aspects of motivational and informational measurement.

Restriction on informational measurement
However, this combined point of view gets a raw deal in the related literature, where particularly informational software measurement is described. The reason behind that is very simple: the work products necessary for the computations of motivational software measurement require a high degree of human creativity, which could be too easily negatively affected by being measured or by being forced to deliver private performance data. As long as the data lead to reward of staff there is no danger, but the other way round it cannot be applied without fear. Goodman [Goo04, p. 9] spots on that: "The first time that a manager uses data supplied by an individual against that individual is the last time that the manager will get accurate or true data from that person."

So, most often a restriction on informational measurement applies in industrial software engineering environments and in the reporting literature. For instance, Pfleeger [Pfl95b, p. 45] restricts her view of meaningful software measurement to process assessment and improvement: "However, measurement is useful only in the larger context of process assessment and improvement. Moreover, choosing metrics, collecting data, analyzing the results and taking appropriate action require time and resources; these activities make sense only if they are directed at specific improvement goals." But Pfleeger's point of view does not reflect the whole story. Hetzel [Het93] states that a certain amount of activities in software engineering potentially profit from software measurement and Fenton [Fen91] suggests that the basic uses of software measurement fall at least into the two broad categories of assessment and prediction. In the reviewed literature, this categorization and the level of detailed background examination of uses and their virtual provenience is subject to the authors' comprehension. But when synchronizing the notes of Humphrey [Hum89], Ejiogu [Eji91], DeMarco [DeM95], Oman and Pfleeger [OP96], Fenton and Pfleeger [FP97], Pfleeger [Pfl97], Whitmire [Whi97], MacDonnell [MSS97], de Champeaux [dC97], Briand et al. [BDR96] [BMB02], Garcia et. al. [GBC+06] the corresponding uses of informational measurement in software engineering spread on:

- *Understanding* the processes and their outcomes in the current situation, establish baselines or objectives to set future behavior.
- *Assessing / evaluating* some product, process activity, or resource to see if it meets predefined (most often quality-related) acceptance criteria like those of laws, standards, project-specific goals, constraints, or customer requirements.
- *Monitoring / controlling* what actually happens in projects to make changes to products, processes, and resources to meet the goals set.

- *Investigation / experimentation* to support or refute a hypotheses or an experiment in terms of the most appropriate software engineering methodologies or technologies.
- *Predicting* future software product characteristics on the base of measures of existing software products.
- *Estimating* the effort of process activities in terms of the combination of measures of existing products with environmental data producing e.g. rates or trend indicators.
- *Improving* of product, processes, resources.

After all, Barnett [Bar05] recognizes the importance of software measurement to prove regulatory compliance e. g. with respect to the U.S. Sarbanes-Oxley Act.

2.4.3 Concerned audiences and information needs

Commercial software measurement must not degenerate into an end in itself and shall provide the basic data for further analysis and derivation of information to satisfy the needs of different organizational audiences.

Consequently, the cross-section of analyses of targeted consumers of software measurement data in literature [Das92] [PJCK97] [Kue00] [Wes02] [LBK05] yields to these person subgroups:

- Senior or functional management,
- Project management,
- Software engineers,
- Specialist groups (software quality management, marketing, process engineering, software configuration management, ...),
- Customers/users.

Whitmire [Whi97] takes on that and relates different views to different consumers of software measures. He recognizes the fact that in usual software development companies, the points of view spread on the strategic issues for senior or functional management, tactical issues for project management, and technical or even combined exploitation of software measurement data for groups like software engineers, specialist groups, and customers, after all.

Strategic view
Being an executive for an organization is obviously a complex task, which not only requires sure instinct and intuition of the senior management and/or the managing director, but also deep insight into the long-term viability of the entire business with the aid of corporate measures. [Jon01] From the strategic point of view, the different projects that might use different tools and techniques have to be overseen in terms of the entire range of aspects like financial, qualitative, or legal ones. Should certain limits of those aspects exceed, the managing director is demanded to urge the particular project managers to intervene to pave the way for continuous economic success. Because established processes may be subject to alignment or improvement, the strategic view strongly relies on the comparison of software measures of different periods in time to develop trends and possibly derive corrective actions problems in *process* or *resource*

efficiency. At this executive level of information need, summarized and precomputed information are preferred. [MG98] That is, apart from being the base for computation, detailed and raw software measures can be counterproductive. In turn, it can be risky to draw more than trend-setting conclusions from the pooled information reflecting solely a high-level view of the story.

The computation of trends for processes or their peculiarities using mathematical constructs like mean or median with report of variability yields to the class of derived measurement (cf. appendix A.5). Whitmire [Whi97] provides usual examples such as:

- Defect rate (e. g.: delivered defects / function point)
- Maintenance efficiency (e. g.: function points / equivalent full-time person)

Tactical view
At the next lower level of decision-making power, often called junior or project management, the performance of particular projects comes to the fore. These project managers are responsible to animate the codified processes and potentially build their project *plans* with the aid of detailed, historical software measures. Because in commercial software development industry a major part of a project manager's salary consists of premium payment, he or she adamantly explores possible deviations from plans and tries to eliminate or *control* them. Once in a blue moon, a project manager will own up not having planned correctly.

Despite the tactician merely being interested in activity-oriented software measures, derived measurement of processes, products, or resources is used. Whitmire [Whi97] mentions that while software measures of the product entity are for instance used to predict project characteristics for planning, the process entity is quantified for controlling aspects like resource usage. Examples for related software measures are e. g.:

- Planning:
  - Time consumption for each task of the work breakdown structure.
  - Cost of operating resources for a certain project.
- Controlling:
  - Effort-to-date for every activity or the entire software product.
  - Defect data (defect rates, containment effectiveness, correction efficiency, cost of rework).

Technical view
Whereas both of the beforehand described views do rely on precomputed, trend-setting data, the technical point of view is either way detached from pure commercial interests and deals with detailed software measures needed for engineering purposes. The measures are reactive and strongly depend on the technology the software engineer uses for the production of the software or of intermediate artifacts.

Being the fundament for all other views, these measures affect technical decisions e. g. regarding requirement acceptance, appropriateness of the design, complexity of algorithms, and so forth. Whitmire [Whi97] illustrates the view with examples, from which some are listed beneath:

- Coupling and cohesion for assessment of the appropriateness of high-level software architecture design.
- Size and complexity for the selection of better, alternative algorithms for software implementation.

2.5 Software measurement paradigms

As should be undoubtedly clear from common sense, multi-faceted software measurement of the complete range of available characteristics asks too much of any organization and does not necessarily make sense. Over the time an immense and not accurately quantifiable number of measures has been proposed in the literature: For instance Zuse [Zus98] reports more than 1,500 and Fenton and Neil [FN00] count about 1,000 general software measures; restricted to the area of object-oriented software measures, Purao and Vaishnavi [PV03] determined a count of more than 370. An early citation of Hetzel [Het93, p. 26] makes clear the problem: "Since we cannot do everything at once, we need some practical guides for prioritizing and selecting the measures we will provide."

Indeed, researchers and practitioners have been trying to address the topic and developed different methodologies to reduce the amount of characteristics to be measured. [Dum05] Today, there are three basic philosophies available: First, there is the *top-down* and/or goal-oriented approach that is aimed at planning and executing the data collection for a lean but most expressive set of software measures. Second, it coexists with the *bottom-up* and/or fixed set of measures approach, which aims at discovering useful information from a set of existing, legacy software measures to allow shaping, attainment, or alignment of organizational goals based on statistical analyses. After all, a *mix* between both paradigms is conceivable, too.

Lassenius [Las06] adds Statistical Process Control (SPC) to the measurement paradigms. This point of view is not shared because all of the introduced paradigms can be utilized to obtain base measurement data necessary for SPC. Furthermore, SPC can solely be meaningfully applied in organizations with repeatable software development processes in place. Although that special topic is worth being investigated further, beyond doubt, for gaining both, brief and in-depth, knowledge the reader is referred to our technical report [DBB+06a] or to other pertinent literature. [BO96] [FPC97] [FC99] [Wel00] [Pan03]

Independent from the software measurement paradigm, that is, whether measurement goals exist or not, a *measurement outline* (true to the original a 'metrics plan') [FP97] should be produced before data collection that gives answers to the important questions of:

1. *Why* shall be measured? (Measurement goals G)
2. *What* entities and attributes shall be quantified by what software measures? (Entities E, attributes A, software measures M)
3. *Where and when* shall the measurements be made during the software process? (Time and frequency as part of operational definitions O)
4. *How* shall be measured, that is, with the help of which tools, collection and analyses techniques, and *who* shall perform it? (Directives D and responsible personnel R)

In accordance with the formal description methodology of Dumke et al. [DSZ05] the set of entities of a measurement outline (M_{MO} or short MO) shall be defined as:

$$\mathbf{MO} = \{G, E, A, M, O, D, R\} \tag{2.5}$$

2.5.1 The top-down approach

The top-down, also called goal-oriented measurement, approach is grounded in early intellectual efforts of US-American researchers around Basili [BZM77] to breakdown the abstract concept 'quality' into a set of factors and sub-factors (criteria) for later quantification (metric). This was the hour of birth for the Factor-Criteria-Metric (FCM) framework. As an intended result, those factors were initially proposed by McCall [MRW77] and subsequently seized again by Evans [EM87] or Deutsch and Willis [DW88]. However, these efforts did lead to a more generalized documentation of the commonsense, threepart principle (cf. figure 2.1) to decompose abstract concepts into measurable factors as a by-product: At first the *measurement goals* have to be established and stated formally on a conceptual level in order to then develop a list of *questions of interest* on an operational level that need to be answered to meet the documented goals. Finally, on a quantitative level, *measures* (true to original called 'metrics') that support or deliver the quantitative answers for the acquainted questions have to be found. Apart from the FCM, the probably best-known, academic top-down approaches are the Quality Function Deployment (QFD) of Kogure and Akao [KA83] and the Goal-Question-Metric (GQM) of Basili and colleagues [BW84] [BR88] [BCR94a] from the University of Maryland.

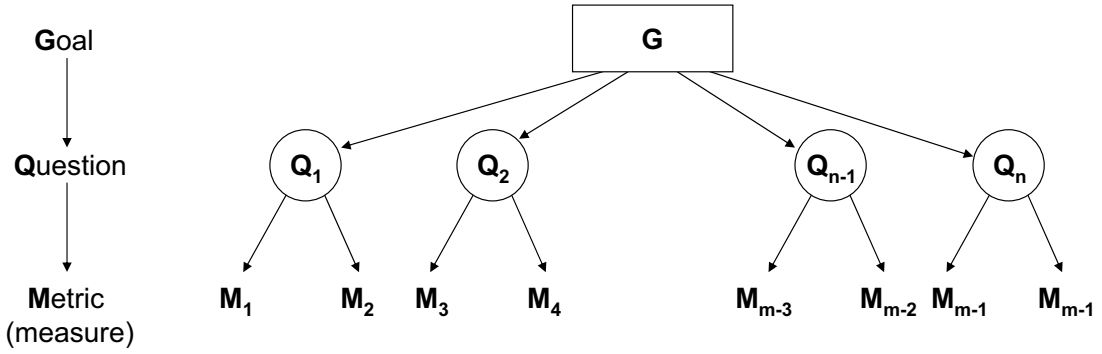

Figure 2.1: An exemplified tree of the GQM model

Early operating experience especially with the GQM has shown that it is superior compared with no systematics at all or any informal, ad hoc approaches, when resources are largely constrained and expert knowledge in the application of software measurement has been established. [RL01] Although being a relatively straightforward method, there have been also many legitimate complaints. For instance Card [Car93] argues: "Despite its popularity as a measurement tool, the GQM paradigm is really better used as a brainstorming technique."

Accordingly, several researchers [Het93] [She95] [BN95] [RL01] provide staggering lists of conceptual problems of the GQM. They mention problems being inherent in top-down approaches as e. g. the need for defining the top. At this point a chicken-and-egg problem may occur, because organizations often strongly depend on reliable (software) measurement to set their goals at the top. Another psychological problem might arise, when practitioners feel that they are not involved in defining and/or interpreting the measures they have to implement. Connected with the latter problem, a determined goal of management at the top might lead to dysfunctional behavior among practitioners like 'finagling' the data to meet the goal, because of the phenomenon that "No one enjoys bearing bad news either, so it gets softened without real intent to deceive." [Bro75] Furthermore, there is the danger immanent in hypotheses-theories to leave out important aspects of the phenomena under study. [RL01] Moreover, it is often

neglected that the software measures materialized once, should evolve as the underlying software development processes evolve to prevent from 'metric monsters' [ZZW95]. Also the number of questions to ask for a certain goal might be unlimited and difficult to quantify with software measures. Lassenius [Las06] mentions another important aspect. Despite of the fact that the behavior of starting the GQM process from scratch every time, a new project comes to the fore, might produce best results for every distinct situation, but, with respect to limited resources, a reuse of developed measures is preferred. The list of issues can be arbitrarily enhanced.

Addressing specifically the lack of process support for the GQM method to some extent, researchers have been trying to mend its basic concepts: Gresse et al. [GHW95] provide an enhanced process model, Briand et al. [BDR96] introduce abstraction sheets, and van Solingen and Berghout [vSB99] document the principles in fine-grained steps. As part of their research Park et al. [PGF96] enhanced GQM's concepts and suggested an advanced Goal-Question-(Indicator-)Measure (GQ(I)M) model. Be it as it may, today the GQM method has been continuously refined and proven to be useful by both, researchers [BvSJ98] [vSB01] [BMB02] [Sch02] [BJ06] [Men97] [MBBD98] [MB00] and practitioners [Lav00] [Kil01] [BH03a].

All things considered, the concept of goal-orientation in measurement is also a very important and common issue in performance measurement of businesses, in general. With respect to the topic of software measurement paradigms and finding the right software measures and/or at least their categories, Herbsleb et al. [HCR+94, p. 53] observe: "Approaches that have worked well elsewhere may also be good candidates for software organizations." In performance measurement multiple approaches exist. Among them range e.g. the 'Three Levels of Performance' by Rummler et al. [RB95], 'Performance Pyramid' by Lynch et al. [LC95] or the Balanced Scorecard (BSc) approach by Kaplan and Norton [KN92] [KN93]. The latter one is aimed at the quantification and controlling of an organization's performance with the kind of performance being dependent on its strategic alignment. Kaplan and Norton propose a balanced picture of four categories of general measures to track: (1) financial, (2) customer satisfaction, (3) internal processes, and (4) innovation and improvement processes. Especially this approach, having the potential to prevent from lopsided points of view, can be simply used and combined with the GQM for software organizations. [BB99]

2.5.2 The bottom-up approach

An alternative measurement philosophy especially for the introduction of software measurement in an organization is the bottom-up and/or fixed set of measures approach. As a kind of 'rapid measurement prototyping' [Buc90a] [Buc90b] it is, backed up by the author's own experience, widely spread in the mental horizon of software engineers and commonly used due to its ability to be a start of software measurement that can be run on a shoestring. It is less frequently documented or cited in software measurement literature, but put forward as the only meaningful starting point to software measurement by proponents like Hetzel and Silver [Het93], Bache and Neil [BN95], or Fuchs [Fuc95]. Moreover, Brown [Bro96] reports that even the Airlie Software Council members like Victor Basili, Grady Booch, Tom DeMarco, or Roger Pressman also recommend the use of this approach.

The suggested procedure was coined 'bottom-up' [Fuc95] or 'Metric-Question-Goal' (MQG) [BN95], because at the time of software measurement it might not be completely obvious why it is done or how to interpret the data. However, experiences of numerous failed software measurement initiatives in industrial settings [Car93] have shown that a top-down paradigm, such as the GQM approach, could not lead to success because no one knew, or could agree upon a measurement goal at the top. Conversely,

organizations were in need to analyze the already existing basic software measures captured in legacy systems to extract their goals, as a start. [Het93] Following the idea of the 'bottom-up' paradigm two scenarios are thinkable:

- On the one hand, one could apply a brute force solution, that stores and/or integrates any legacy, genuine data available in a single system. Based on complex analyses [Men97] of the raw data, a set of software measures can be defined **a posteriori**. Although storage and analyses of a maximum of data would most probably exceed the budget and technical feasibility in any organization, these efforts do not promise an added value equivalent to the expenses. More specifically, there is "the danger of having too much data with the likelihood of systematic and directed analysis much reduced." [RL01] However, a limitation of the amount of legacy, genuine data to regular snapshots could offer a meaningful trade-off.

- On the other hand, an **a priori** definition of a fixed set of primary software measures, on which secondary software measures could be computed (cf. appendix, section A.5) can promise success. Hetzel and Silver [Het93] argument that this set should focus on the fundamental objects of software engineering, that is, on the work products and resources used to create them. It is then to be collected on every software (intermediate) product and should cover the vector of inputs (such as resources or activities), outputs (the technical work products), and results (usage and effectiveness of the work products according to the requirements. The interpretation of the data then can stimulate the development of questions that may lead to the setting and/or realignment or of organizational goals or to corrective actions for goal-attainment. [Fuc95]

  When utilizing the a priori bottom-up approach, the challenge of finding a suitable set of well-established software measures has to be coped with: For instance, Rubin proposes ten categories of software measures to build a 'universe'. [Rub93] Furthermore, driven by the need of the U. S. Department of Defense (DoD) to evaluate their software acquisitions, a set of core software measures has been assembled by the commissioned SEI in 1992. [CPG+92] This set of measures for characteristics of *size, effort, schedule, quality*, and even *rework* the DoD required to be collected by their contractors and thus became a de-facto standard. [CPG94] [Hei01] Other well-known proponents of fixed-sets of software measures are e.g. Putnam and Myers [PM97]. In general they [PM03, p. 33] suggest software measures "that enable software development to operate successfully in a market economic system." Within their book "Five Core Measures" they discuss and recommend the following five core derived measures that fit best for the tactical view: quantity of function (*size*), *productivity*, *time*, *effort*, *reliability*.

In particular, and based on profound understanding of the processes at hand, the MQG method as adumbrated in figure 2.2, should start with the collection of all legacy data or a respective set of *software measurement results and analyzing* them. Afterwards, *questions should be asked* in order to feel out why results are the way they are and, whether there is demand for improvement to reduce deviations from expectations. As a result, people in charge of pertinent management positions are aware of possible problems and can *lay out corrective actions or set goals*, eventually.

When performing a cost-benefit analysis, the factor of effort and cost reduction due to having a fixed set of predefined measures at hand cannot be dismissed. Beyond the saved effort for not being forced to redevelop from scratch arbitrary measures every time a new software development project comes up, also the question of tool support has to be answered only once. Due to large involvement and direct benefits

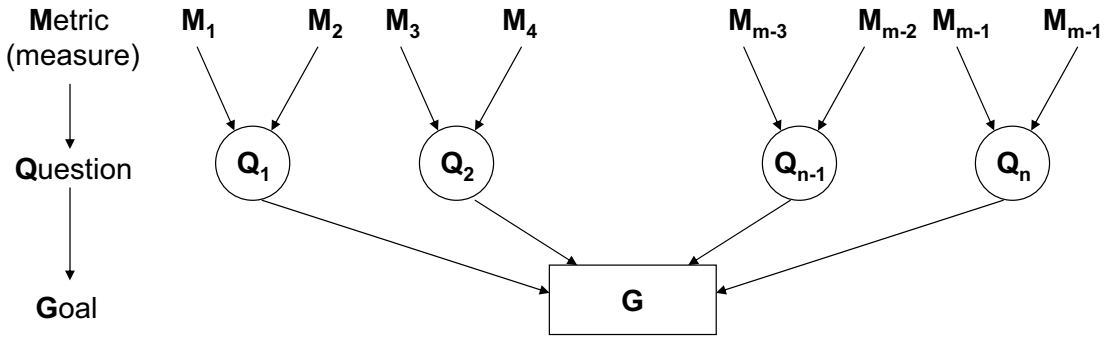

Figure 2.2: An exemplified tree of the MQG method

for practitioners, this approach is likely to redeem from dysfunctional behavior caused by fear. Moreover, this philosophy promotes asking questions and having raw data to answer nearly all upcoming, technical questions. However, the old phrase holds true: "What you gain on the swings, you loose on the roundabouts." Because the described philosophy assumes that the characteristics of all software projects are similar, this approach can be problematic since the opposite holds. [McG01] Even if, in the best case, certain factors might match, there will most probably remain factors that make lumping together error-prone. While this paradigm for software measurement is a sublime starting point in general, it should be applied carefully to prevent 'data cemeteries' [DFS93] [Deb00] and thus very soon sophisticated towards the top-down approach in order not to "... cause a measurement process to fail." [Hol02]

2.5.3 The mixed approach

As its label implies, the mixed approach is an intermediate step on the way from the application of the bottom-up to the top-down paradigm. As a hybrid it has the potential to combine elements of both philosophies to provide an effective, efficient, and feasible evolutionary step. Ramil et al. [RL01] summarize the synergy of both in three steps:

1. The application of the top-down paradigm (e. g. using the GQM or BSc) will lead to a minimum but expressive set of software measures.
2. Then, the application of the bottom-up paradigm will lead to the storage of a certain amount of legacy, genuine data.
3. After all, it is to be verified by analyses that the set of software measures materialized in step one can by entirely derived from the legacy, genuine data of step two. If this were not the case, missing information were to be added to step two.

2.6 Synthesis of elements: The software measurement system

Summarizing the findings of appendix A and chapter 2, the synthesis of the described elements forms a software engineering measurement system (SMS). With respect to the formal representation of an exemplar measurement system by Dumke et al. [DSZ05, p. 31] as well as the remarks with collaboration of the author of this thesis [DBK+06], it has been slightly modified by the author in order to be consistent with the argumentation within this thesis. Thus, it shall be defined as:

$$\begin{aligned} \textbf{SMS} &= (M_{SMS}, R_{SMS}) \\ &= (\{MO\} \cup \{Q, I\}, R_{SMS}) \\ &= (\{G, E, A, M, O, D, R, Q, I\}, R_{SMS}) \end{aligned} \tag{2.6}$$

For better comprehensibility, the description and as well as the reference in brackets of the respective constituents of the set of entities, M_{SMS}, is given in table 2.4 putting up some redundancy to the information given in the sections before.

G	=	The possibly empty set of business *goals* and/or measurement information needs from the empirical relative being the driver behind software measurement. [Refer to sections A.3 and 2.4]
E	=	The non-empty set of software engineering *entities* to be measured. (In the complete case $E = \{SD, SP, SR\} = \{SDP\}$) [Refer to section 2.3]
A	=	The non-empty set of *attributes* of the software engineering entities to be measured. [Refer to section 2.3]
M	=	The non-empty set of software *measures* representing the mapping rule from the empirical relative to the numerical relative. [Refer to section A.4]
O	=	The non-empty set of *operational definitions* such as points in time or frequencies for collecting and analyzing the data. [Refer to section 2.5]
D	=	The non-empty set of measurement *directives* including data collection and analysis. [Refer to sections A.6 and 2.5 and to the author's technical report [BKD05]]
R	=	The non-empty set of *responsible* personnel for data collecting and analysis. [Refer to section 2.5]
Q	=	The non-empty set of *quantities* and/or formal objects (numbers, symbols, or structures). [Refer to section A.3]
I	=	The possibly empty set of *information products*. These can be represented by the cartesian product of software measures (values V with $V \subseteq Q$) and units (U) as $V \times U \longrightarrow I$. [Refer to section A.6]

Table 2.4: Elements of a software engineering measurement system

2.7 Software measurement process models

Following the notes of McAndrews [McA93] software measurement must support and hence be aligned with the overall software processes of an organization. Hence and based on the notes of Liptak [Lip92] and McAndrews, the following is postulated:

> **Definition 2.7:** The *software measurement process* is a certain portion of an organization's software engineering process that provides for the identification, definition, collection, storage, and analysis of software measures of used, consumed, or produced entities of the software process.

Suggested by the above definition, the general software measurement process entails more than just the core of involving the rule-based and objective assignment of numbers or symbols to attributes of software engineering entities. Subject to the chosen, particular measurement paradigm a number of steps has to be performed. For instance, the data gained by software measurement has to be stored and analyzed to derive information. For software measurement to be performed by individuals, a guideline consisting of the very coarse activities, that come under the umbrella of the software measurement process, should be declared in a conceptional and/or graphical form. This is being done with the aid of software measurement process models. It is common among all these models that they are based on the Quality Circle 'Plan-Do-Study-Act' (PDSA) of Shewhart [She31] dating back to the late 1920s.

Taking a SMS for granted as represented in equation 2.6, with respect to a similar, formal definition of Dumke et al. [DSZ05], the software measurement process (MP) shall be formally defined a certain instantiation of a SMS:

$$\begin{aligned} \mathbf{MP} &= (G \times E)_{SMS} \longrightarrow_{P_{materialization}} MO_{SMS} \\ &\quad \longrightarrow_{P_{collection}} Q_{SMS} \longrightarrow_{P_{analysis}} I_{SMS} \longrightarrow_{P_{exploitation}} G''_{SMS} \end{aligned} \tag{2.7}$$

$$\begin{aligned} &= (G \times E)_{SMS} \longrightarrow_{P_{materialization}} (G' \times E' \times A \times M \times O \times D \times R)_{SMS} \\ &\quad \longrightarrow_{P_{collection}} Q_{SMS} \longrightarrow_{P_{analysis}} I_{SMS} \longrightarrow_{P_{exploitation}} G''_{SMS} \end{aligned}$$

(G' extends G trough more refinement relating to E and A, E' extends E by modeling or abstractions, and G'' extends G' by exploitation of obtained measurement information products and experience.)

The above equation embodies the materialization of (possibly non-existent) goals (G) for an entity of interest (E) under certain starting constraints using a subset of measurement procedures ($P_{materialization}$). These can either be special goal materialization techniques or elicitation of predefined sets of software measures. The transformation leads to a measurement outline (MO), which consists of the singular constituents as described in section 2.5, and more tightened constraints such as cost or elevated ones as e. g. the experience of measurement personnel. On the base of the measurement outline and the constraints, data collection ($P_{collection}$) can be performed. This results in quantities (Q), which can be analyzed using analysis procedures ($P_{analysis}$) and even more interfered constraints. Once, the analysis has produced the measurement information products (I) they can be exploited together with procedures ($P_{exploitation}$) and constraints to either set measurement goals for the first time or to sharpen them (G'').

Furthermore, equation 2.7 provides the foundation for the formalization of measurement-based software engineering product improvement: For the context of this thesis it shall be defined as the application (use U) of the software measurement process (MP) in the following manner:

$$\begin{aligned}\mathbf{U}(\mathbf{MP}) \quad = \quad & (G_{improvement} \times E)_{SMS} \longrightarrow_{P_{materialization}} MO_{SMS} \\ & \longrightarrow_{P_{collection}} Q_{SMS} \longrightarrow_{P_{analysis}} I_{SMS} \longrightarrow_{P_{improvement}} E_{SMS}^{improved}\end{aligned} \tag{2.8}$$

with $G_{improvement} = \{increaseQuality, decreaseCost, \ldots\}$ being the set of goals for measurement-based improvement and $P_{improvement}$ being the procedures and/or corrective actions to the software development life cycle based on the software measures.

The dominance of top-down oriented software measurement process models
As mentioned before, over the time the most flexible but at the same time most complex, top-down measurement paradigm together with the related GQM method have proven to be superior in a majority of cases yielding to a *de facto* autarchy. [Fuc95] Despite being a sophisticated concept, in general, the application of an entire software measurement methodology based on the top-down measurement paradigm for industrial purposes requires concrete support concerning process guidance, in particular. That demand did not die away unheard and has been addressed multiply by both, practitioners and/or researchers: In the first place, the top-down paradigm could find proliferation due to those supportive software measurement process models. [BvSJ98]

Hence, this section presents the most prevalent measurement process models following the top-down paradigm. Up to date, to the knowledge of the author, there are neither software measurement process models for the bottom-up nor for the mixed paradigm publicly available.

2.7.1 The ami measurement process model

Already in 1992 a project called application of metrics in industry (ami), sponsored by the Commission of the European Communities via the European Strategic Programme for Research and Development in Information Technology (ESPRIT), culminated in a method and/or process model of the same name, ami. [KCCHS92]

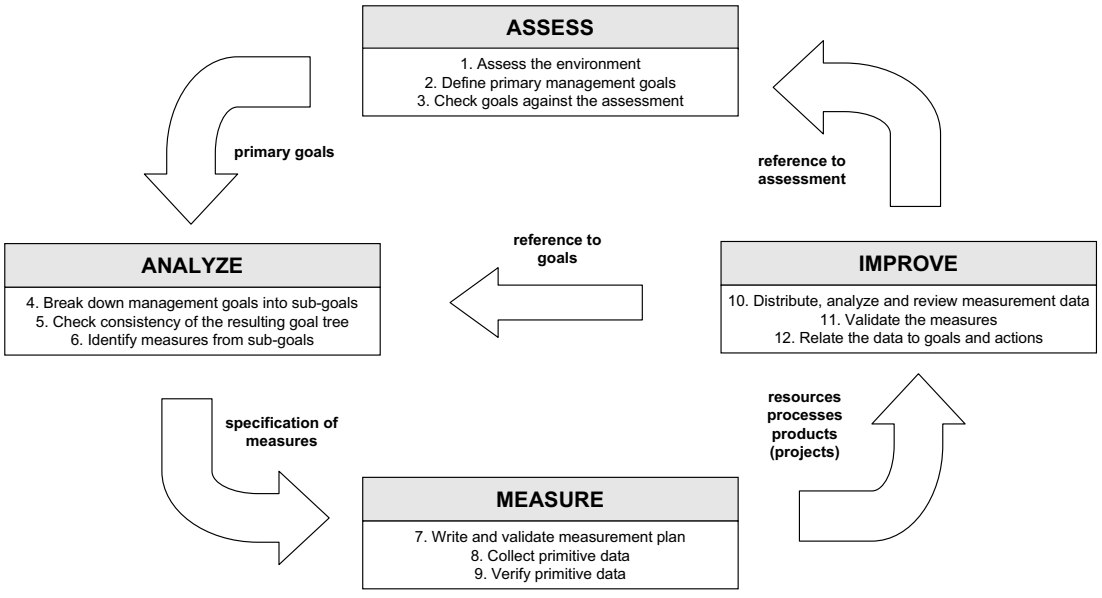

Figure 2.3: The glue between the ami method's four activities and 12 steps

The elaboration of the method was backed up by full-scale trials in practice driven by the developing consortium that consisted of experts and users of software measurement. [PKCS95] Starting again from the large-grained four-step PDSA cycle, the circular ami software measurement process model intends to serve all audiences as illustrated in figure 2.3 and consists of four distinct activities with each having three own tasks to be fulfilled. The cycle starts with a dedicated set of improvement goals $G_{improvement}$ for software engineering entities E and should end with them being improved (E').

2.7.2 The general Jacquet and Abran model

Contributing to a better understanding, Jacquet and Abran [JA97] provide a high-level model as overview and a more detailed model of the (software) measurement process. The authors recognized the need for a formal methodology as already argued for by others like Gustafson et al. [GTW93] and thus structured the process with their high-level model into four activities: (1) Design of the measurement method; (2) Application of the measurement method rules; (3) Analysis of the measurement results; and (4) Exploitation of the measurement results. Obviously, the existence of measurement goals (G) is presumed and the exploitation step deals with the virtual exploitation of the software measures rather than with sharpening the measurement goals (G'').

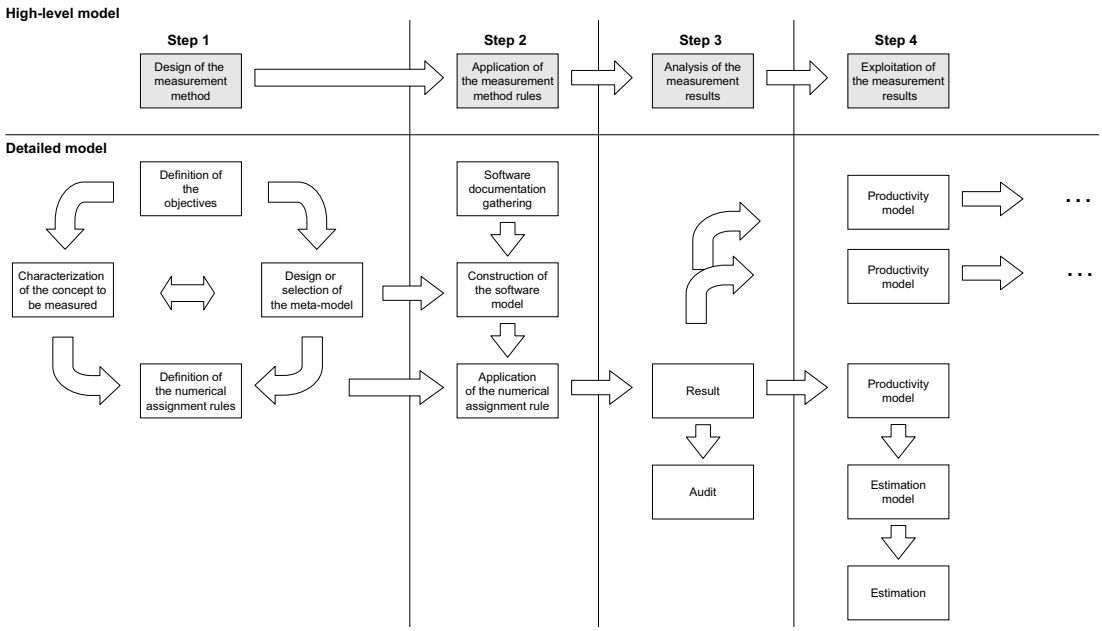

Figure 2.4: High-level and detailed models of the software measurement process (adapted from [JA97, p. 130])

While for the design and the application of the measurement method's rules *structure models* can be used, for the analysis and exploitation of the software measurement results entity population models are available for disposal. [KPF95] As can also be recognized in figure 2.4 in the activities, detailed models have a right to exist, in turn. These detailed models involve a number of specific tasks to be fulfilled.

2.7.3 The GQM-based measurement process of van Solingen et al.

After the GQM process model of Gresse et al. [GHW95], in an additional endeavor to enrich the GQM with an especially compiled, supporting process model for application in industry, van Solingen and Berghout [vSB99] compiled steps and procedures contributing to software measurement following the top-down paradigm. They arrange these steps around the four different phases of (1) Planning; (2) Definition; (3) Data collection; and (4) Interpretation as depicted in figure 2.5. As the denomination of the measurement process model implies, the existence of measurement goals (G) is a requirement for using that model.

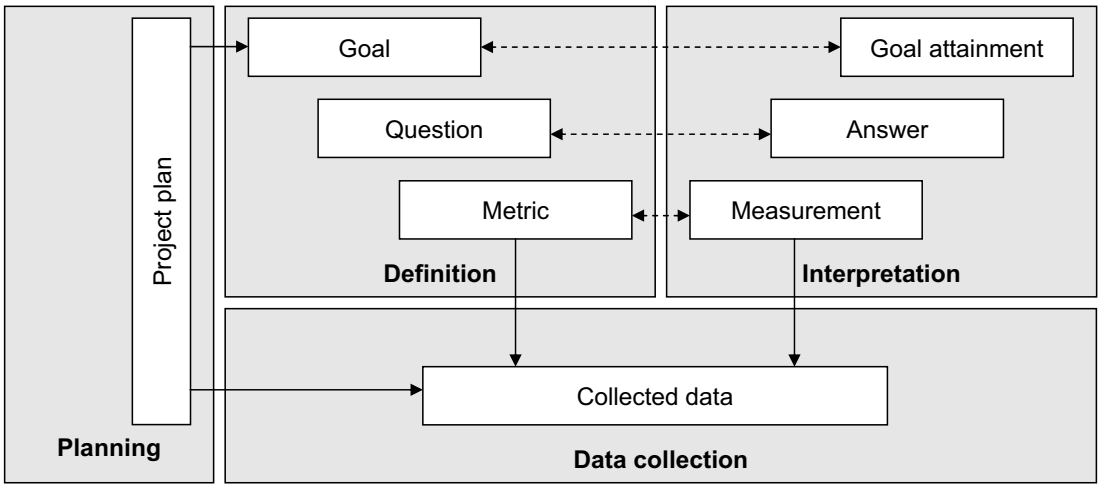

Figure 2.5: The GQM method of van Solingen and Berghout (adapted from [vSB99, p. 22])

The basic steps and procedures that constitute the respective phases of their subjective interpretation of the GQM are as following [vSB99]:

1. "Planning
   (a) Establish GQM team.
   (b) Select improvement area.
   (c) Select application project.
   (d) Create project plan.
   (e) Training and promotion.
2. Definition
   (a) Define measurement goals.
   (b) Review or produce software process models.
   (c) Conduct GQM interviews.
   (d) Define questions and hypotheses.
   (e) Review questions and hypotheses.
   (f) Define metrics.
   (g) Check metrics on consistency and completeness.
   (h) Produce GQM plan.

(i) Produce measurement plan.

(j) Produce analysis plan.

(k) Review plans.

3. Data collection

(a) Hold trial period.

(b) Hold kick-off session.

(c) Create metrics base.

(d) Collect and check data collection forms.

(e) Store measurement data in metrics base.

(f) Define analysis sheets and presentation slides.

4. Interpretation

(a) Prepare feedback session.

i. Update the analysis sheets.

ii. Create additional material.

iii. Update presentation slides.

iv. Review presentation slides.

v. Save copies.

vi. Create & distribute handouts.

(b) Organize and hold feedback session.

(c) Report measurement results."

2.7.4 The PSM (and ISO/IEC Standard 15939) process model

Having notice of the fact that an organization's performance essentially depends on the economic success of its projects, the creators of the PSM guidebook — headed up by the luminary McGarry [MCJ+01] — intended to address predominantly information needs and characteristics at the project entity level (cf. section 2.3.4). They provide a common measurement process model as well as a common measurement information model. Because the PSM guidebook's principles and models represent subject-matter agreements among international experts, the ISO/IEC standardization bodies have accepted the models of PSM as standard 15939 [ISO02] in 2002. Moreover, PSM has become the intellectual foundation [Jon03a] for the 'Measurement and Analysis' support process area of SEI's CMMI. [SEI02a, SEI02b]

With the measurement process model being of elevated importance it shall be stressed in the context of this thesis. The model's version as manifested in ISO/IEC Standard 15939:2002 is therefore illustrated in figure 2.6. Once more in accordance with the PDSA cycle, the four process categories 'establish & sustain measurement commitment', 'plan the measurement process', 'perform the measurement process', and 'evaluate measurement' go into the model together with a number of tasks to be fulfilled having a 'technical or management process' as initiator and concerned audience. [DBH05] Thus, this enhanced process model extends the software measurement process (MP) as defined in equation 2.7 by preparatory activities to establish and sustain commitment ($MP_{Committment}(MP)$) and following-up evaluation activities ($MP_{Evaluation}(MP)$) of the software measurement process.

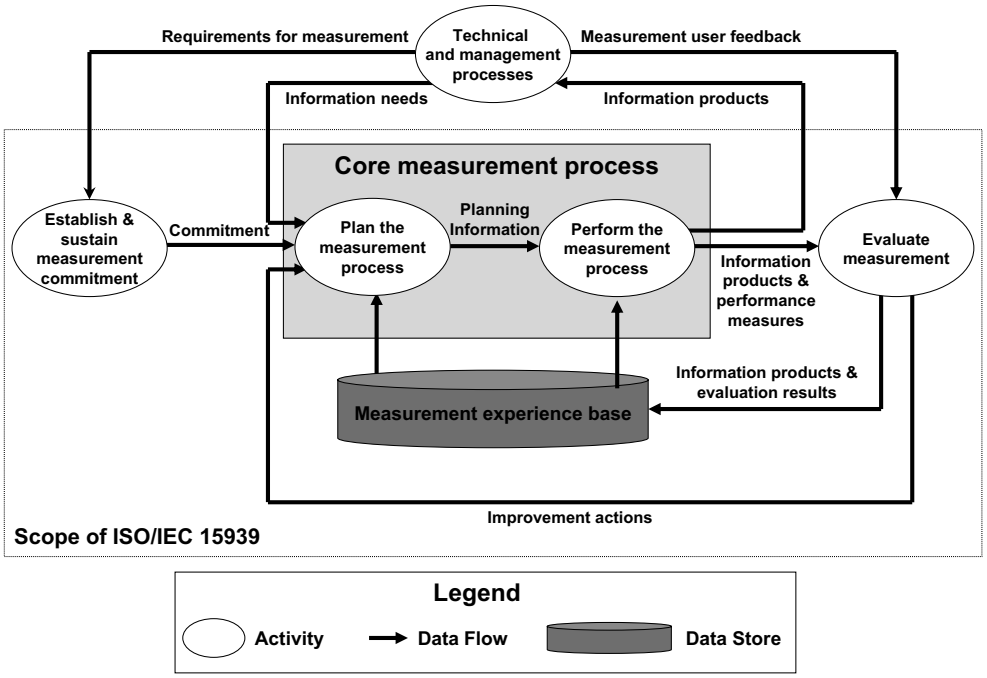

Figure 2.6: The ISO/IEC 15939:2002 measurement process model (taken from the standard)

The pertinent activities are as per the standard:

1. **"Establish and sustain measurement commitment**

   (a) Accept the requirements for measurement

      i. The scope of measurement shall be identified.

      ii. Commitment of management and staff to measurement shall be established.

      iii. Commitment shall be communicated to the organizational unit.

   (b) Assign resources

      i. Individuals shall be assigned responsibility for the measurement process within the organizational unit.

      ii. The assigned individuals shall be provided with resources to plan the measurement process.

2. **Plan the measurement process**

   (a) Characterize organizational unit

      i. Characteristics of the organizational unit that are relevant to selecting measures and interpreting the information products shall be explicitly described.

   (b) Identify information needs

      i. Information needs for measurement shall be identified.

      ii. The identified information needs shall be prioritized.

      iii. Information needs to be addressed shall be selected.

      iv. Selected information needs shall be documented and communicated.

(c) Select measures

i. Candidate measures that satisfy the selected information needs shall be identified.

ii. Measures shall be selected from the candidate measures.

iii. Selected measures shall be documented by their name, the unit of measurement, their formal definition, the method of data collection, and their link to the information needs.

(d) Define data collection, analysis, and reporting procedures

i. Procedures for data collection, including storage and verification shall be defined.

ii. Procedures for data analysis and reporting of information products shall be defined.

iii. Configuration management procedures shall be defined.

(e) Define criteria for evaluating the information products and the measurement process

i. Criteria for evaluating information products shall be defined.

ii. Criteria for evaluating the measurement process shall be defined.

(f) Review, approve, and provide resources for measurement tasks

i. The results of measurement planning shall be reviewed and approved.

ii. Resources shall be made available for implementing the planned measurement tasks.

(g) Acquire and deploy supporting technologies

i. Available supporting technologies shall be evaluated and appropriate ones selected.

ii. The selected supporting technologies shall be acquired and deployed

3. **Perform the measurement process**

(a) Integrate procedures

i. Data generation and collection shall be integrated into the relevant processes.

ii. The integrated data collection procedures shall be communicated to the data providers.

iii. Data analysis and reporting shall be integrated into the relevant processes.

(b) Collect data

i. Data shall be collected.

ii. The collected data shall be stored, including any context information necessary to verify, understand, or evaluate the data.

iii. The collected data shall be verified.

(c) Analyze data and develop information products

i. The collected data shall be analyzed.

ii. The data analysis results shall be interpreted.

iii. The information products shall be reviewed.

(d) Communicate results

i. The information products shall be documented.
ii. The information products shall be communicated to the measurement users.

4. **Evaluate measurement**

(a) Evaluate information products and the measurement process
i. The information products shall be evaluated against the specified evaluation criteria and conclusions on strengths and weaknesses of the information products drawn.
ii. The measurement process shall be evaluated against the specified evaluation criteria and conclusions on strengths and weaknesses of the measurement process drawn.
iii. Lessons learned from the evaluation shall be stored in the 'Measurement Experience Base'.

(b) Identify potential improvements
i. Potential improvements to the information products shall be identified.
ii. Potential improvements to the measurement process shall be identified.
iii. Potential improvements shall be communicated."

2.7.5 Other top-down measurement process models

Similar, but less frequently utilized approaches to promote the acceptance of goal-oriented software measurement in industry from a rather scientific perspective are e. g. reported from Paulish and Möller [PM92], McAndrews [McA93], van Solingen and Berghout [vSB01] or Dumke et al. [Dum03] [EDBS05].

2.8 Software measurement programs

Translating the theoretical foundations, concepts and the exemplary software measurement process into practice of software engineering industry has been a demand of progress and thus a favored topic of publications for practitioners and researchers as can be recognized from the number of references examined. In this vein, the virtual application of software measurement in commercial software engineering settings for the main purpose of SPI has been suggested to be established in the course of *Software Measurement Programs*, for which the denominations 'metrication programme' [Kit96b], 'metrics program' [HG98] [PPS03] [Goo04], or 'measurement framework' [Men97] hold, too. Liptak [Lip92], Abran et al. [ALB99], or Berry and Jeffery [BJ00] similarly define:

> **Definition 2.8:** "A *software measurement program* is the set of on-going organizational processes required to define, design, construct, implement, operate and maintain an information system for collecting, analyzing and communicating measures of software processes, products and services."

Having in mind equations 2.1, 2.2, 2.3, and 2.5, a SMP shall be formally described as shown in equation 2.7. It is worth to be noted that the entities process (SD), product (SP), and resource (SR) might act as both, as an entity of interest to be quantified as well as a building block of the SMP itself.

$$\begin{aligned} \mathbf{SMP} &= (M_{SP}, R_{SMP}) \\ &= (M_{SDP} \cup M_{SMS}, R_{SMP}) \\ &= (M_{SD} \cup M_{SP} \cup M_{SR} \cup M_{SMS}, R_{SMP}) \end{aligned} \tag{2.9}$$

2.8.1 Examination of the current situation

When examining the current situation as it is reflected by the reference material of the technical literature [KLBD06] [KBD06] [WBD06], one can receive the impression that, across the board, the companies having embarked on SMPs could succeed with their attempts. The literature is for instance rife with an impressive number of often cited publications of Grady [GC87] [Gra92] [Gra94], who describes — not without pride — how HP Inc. could successfully establish such an initiative that he used to be the lead of. Reports of predominantly major players and/or companies such as the U. S. Army [Fen90] [Pau93a], Motorola [Das92], Contel [Pfl93], Bull Honeywell [Wel94], Eastman Kodak Company [Sed97], Nokia [Kil01], or Lockheed Martin [Jon03b] seem to mirror an ideal world of SMP implementation. And even from a few smaller companies like for instance Modern Technologies [Bak91], Loral [NB93], Schlumberger [vLvSO+98], Electronic Data Systems [dS02], Financial Software Solutions [IM00] [IM03], or Sogeti Nederland B. V. [Dek05] and other anonymous ones [Kau99], measurement program successes are reported.

However, the real situation with SMPs is fairly different and rather disastrous than optimistic: According to Kaner and Bond [KB04] just a few companies establish measurement programs and even fewer can finally succeed with it. But those that come to grief are often kept secret. [HF97] As early as from 1988 the researcher and analyst Howard Rubin [Rub90] prepared statistics of SMPs' successes or failures. Since then, these statistics continuously show a consistent failure rate of 78% or higher before two years of the program's successful operation have passed. In addition, in the same study he observes that oblivion of the SMP's exploitation is usually at high 15% and real usage at very low 5%. Also in the 1980s, an industry survey [Het90] financed by Xerox and Software Quality Engineering revealed that not even 10% of the respondents had a positive image of software measurement due to bad own experiences. Another corroborating study of Desharnais [Des94] among twenty Canadian companies analyzed the successes and failures in the implementation of SMPs and exhibits a similar failure rate of 60%. The most recent and multiply cited [Dek99] [DM02] [MD04] of those surveys of Rubin dates back to 1998. It mirrors the even worse situation of nearly 80% failure rate before their second anniversary out of an population of up till then 800 examined SMPs in industry.

2.8.2 Costs and benefits of SMPs

Implementing and sustaining SMPs in industry are most often cost and labor intensive undertakings [Ket06], which organizations do not perform for an end in measurement itself but to generate extra value by deeper insights into their entities of interest. [NvV01] Apart from organizations that do not really have the choice whether to risk implementing and sustaining a SMP or not because of being forced by contract or legal regulation, it usually comes down to a cost-benefit analysis for those organizations being free in their decisions. Needless to say that the return must be worth the investment of financial and personnel resources. [RH96b] [RH96a] [RH97] [Der00] Usually, a cost portion of 2% up to 8% of the annual development or project total has to be accounted

for, when believing in experiences reported in literature. [Gil92] [FN99] [GJPD99] However, this amount strongly depends on contextual aspects of the organization and the desired extent of the SMP. [IFP04] Herbsleb and Grinter [HG98] summarize their cost-related lesson learned in implementing a SMP and alert their audience: "Meaningful organization-wide metrics may be prohibitively expensive, even when they appear simple, and these costs are likely to be grossly underestimated."

As partly listed in table 2.5 benefits of SMP endeavors are often intangible and difficult to express in numbers. [vSB99] In contrast, the cost of not implementing and sustaining it can be determined quantitatively in terms of failing projects or even the entire business.

Item	**Cost**	**Benefit**
Training	Class costs	Future expertise
	Less time on project	Consistency, standardization
Management	More management time	Less management time
Engineering	Start-up costs	Increase productivity
	Data collection time	Reduce development time
	Analysis time	Fewer defects
		Reduce defect find/fix time
		Reduce maintenance
		Increase reusability
		Increase user satisfaction
Capital expenses	Purchase hardware/software	

Table 2.5: Cost versus benefit of SMPs (adapted from [Gil92])

2.8.3 Pitfalls of SMP implementation and sustainment

The reasons, why a large share of SMPs has been discontinued within a short period of time after set-up, are of a complex nature and span more on managerial and cultural than on technical or measure-related issues. [JB93] Accordingly, McQuaid et al. [MD04] argue that "... there is more to measurement than technical implementation." As early as in 1990, Verdugo shared his knowledge on reasons for SMP failures in one of Rubin's early surveys. [Rub90] Although the pitfalls extracted from former experiences with SMP implementation have been pointed out frequently in the meanwhile of the last two decades, nowadays the complaints still range among the road bumps on the way to successful industrial software measurement. Even worse, the issues can now be complemented by a list of extra hazards reported in more recent publications. Apparently, the hitherto drawn consequences have not been sufficient enough to address the emerging issues, which shall be exemplarily summarized as following:

1. Basic misunderstanding of the theory underlying software measurement [Kan00] [DM02]

2. *Absence of verifiable objectives (unmanaged expectations)* of the SMP, measurement overkill, mismatch, or unplanned measurement to meet criteria of SPI normative maturity models [Rub90] [Fen91] [Wie97b] [KB99] [BH03a] [MD02] [Den02] [MD04] [KB04];

3. *Lack of ongoing education*, ignorance, or misunderstanding of the underlying measurement theory among staff [Eji91] [PJCK97] [Zus98] [DB99] [Den02] [DM02] [BH03a] [KB04];

4. *Undervaluation* of the highly complex interrelations between process, products, and people in software engineering by management [RJ94] [Zus98] [ALB99] [DB99] [DM02];
5. *Resistance and fear* of staff against performance monitoring, added effort, and misinterpretation of numbers [Rub90] [Eji91] [Fen91] [Aus96] [DB99] [KB99] [Min00] [MD02] [dS02] [Den02] [MD04];
6. *Management's mislead punishment attempts* for the bearer of bad measurement data reflecting the current state [Aus96] [Hof00] [DM02];
7. *Nonexistence or expensiveness of automated tool support* [Fen91] [BH03a];
8. *Disability to generate measurement-based action /* measurement as an end in itself [Rub90] [Fen91] [dS02] [Den02] [MD04];
9. *Lack or withdrawal of senior management's commitment* yielding insufficient staffing and discontinuation of founding [Rub90] [Fen91] [DB99] [KB99] [Den02];
10. *Omission of extensive communication and advertisement* of goals, procedures and results of the SMP with staff members [DB99] [KB99] [dS02] [Den02] [MD04];
11. *Expectation that measurement be a finite project* causing change automatically. [KB99]

2.8.4 Best practices for SMP implementation and sustainment

Because it lies in the general nature of engineers to learn their lessons from mistakes made once, based on research or on comparisons of operating experiences with SMP implementation attempts, a multitude of authors has published lists of success and risk factors, characteristics of highly successful SMPs, determinants of success, and other measurement-related, recommended activities that call for their compilation towards complete works. But these factors have to be seen with a pinch of salt, too, as Niessink and van Vliet [NvV98] observe: "The success factors for software measurement, though highly useful, do not differ all that much from the hit list for software reuse, formal specifications, or any major organizational change relating to the software process. The give premises for success, not roads to get there." Be it as it may, while many authors [Abr01] [RHB03] [Rif03] [Dyb05] [NWZ06] have published extensive lists of generally applicable factors of which software measurement is a major one, e. g. Gopal et al. [GGM99,GKMG02,GMK05] or Umarji and Emurian [UE05] extracted factors specific to SMPs. In general software engineering, recommended activities that have been proven to yield to success are commonly called *best practices*. [DLW99] Fortunately, the situation is now alike in the area of software measurement, where an evolution from tentative experiments to best practices is taking place. [Gra92] That is the motivation to provide an inventory of SMP implementation best practices as extracted from a voluminous literature survey. Over and above, it should be noted here that the literature on SMP best practices nearly exclusively presumes the top-down paradigm and omits to respond to the bottom-up paradigm. Nevertheless, both categories are dwelled on.

Best practices for SMPs following the top-down paradigm
For a long period of time, at first Fenton [Fen91] and then Jeffery and Berry [JB93] were the only authors, who published a fairly comprehensive list of those success factors being structured according to the few main focal points around the *context*, the *inputs*, the *process*, and the final *product* of a SMP in industry. Later and with the intention to provide an inventory of SMP risk factors, Abran et al. [ALB99] enhanced the

existing list of beneficial aspects (as for instance published by Hall and Fenton [HF97]) by cognitions evolved from their own literature study and proposed a *measurement program implementation reference model*. Their model subdivides the factors in relation to five distinct focal points around the preparation of an implementation *context* for the SMP, the set-up of an adequate *organization* of structures, the program's *components*, its *results*, and its advancement referring to maintenance and *evolution*. Although, Gopal et al. [GKMG02,GMK05] subdived their revealed critical success factors for SMP implementation and sustainment into *technical* and *organizational* factors, the model of Jeffery and Berry yields a sufficient and straightforward level of abstraction, it shall serve as structural element for the description of an updated set of SMP best practices as extracted during this research's literature study. Thus, a subjectively selected subset of the revealed and most often mentioned best practices are logically collocated, assigned to the proposed dimensions in the following list and labeled according to the mentioning publications.

Context-related best practices

- Do research in software measurement theory. [GC87] [JB93] [Dek05]
- Ensure that a quality assurance environment is in place. [JB93]
- Ensure that software process and methodology bear stability (e. g. software process maturity level) [JB93] [She94] [Den02]
- Define and clearly state the objectives of the SMP for different audiences in congruency with the overall business goals and/or organizational triggers. [GC87] [MIO87] [Rub87] [Mil88] [CG90] [Fen91] [JB93] [Rub93] [Kit96b] [OJ97] [KB99] [Kul00] [Min00] [NvV01] [Hol02] [DM02] [Den02] [DN03] [Goo04] [IFP04] [EDBS05] [Dek05] [Dyb05]
- Build a participatory management style involving all levels of audiences. [JB93]
- Ensure a supportive industrial climate of trust, respect, and esteem among all levels of staff with a predisposition to improvement [JB93] [Rus02] [DM02] [IM03]
- Ensure that the level of technical difficulty of the software development process and products is within the capability of staff members. [JB93]
- Identify an internal software measurement champion having the responsibility for SMP implementation and sustainment as well as for obtaining the benefits from the SMP. [Das92] [JB93] [GCWF95] [HF97] [Cla02] [Goo04]
- Provide a realistic assessment of the implementation and sustainment cost and period as well as pay-back period. [GC87] [RC91] [JB93] [Kit96b] [GJPD99] [KB99] [DN03]
- Realign the corporate reward system to promote software measurement and collection of unbiased data. [Pfl93] [DB99] [IM00] [IM03] [Dek05] [UE05]

Input-related best practices

- Convince all involved audiences of the importance and meaningfulness of software measurement to ensure commitment via promotion. [GC87] [MIO87] [Fen91] [JB93] [GCWF95] [Kit96b] [Wie97b] [HF97] [Wie97a] [GGM99] [GJPD99] [KB99] [Kau99] [LR99] [BJ00] [Min00] [Cla02] [GKMG02] [Rus02] [Jon03b] [Dyb05]

- Ensure proper raising of resources by senior management. (Extra tasks require extra resource, not on top.) [GC87] [Fen91] [JB93] [She94] [Wie97a]
- Provide training to software measurement data collectors on how software measurement relates to the problems to be solved. [GC87] [RC91] [Das92] [JB93] [Pfl93] [Kit96b] [Wie97b] [Wie97a] [HF97] [GJPD99] [Kau99] [KB99] [LR99] [NvV01] [DM02] [GKMG02] [Rus02] [Jon03b] [IFP04] [EDBS05] [Dek05]
- Ensure sufficient managerial experience and training of concerned audiences to use the software measurement data. [JB93] [Wie97b] [Wie97a] [Kau99] [LR99] [Jon03b] [DM02] [EDBS05] [Dek05]
- Contemplate external consultants where needed to get additional experience and authority. [JB93] [HF97]
- If possible, make use of existing knowledge and materials of software measurement in the organization. [BR88] [HF97] [Wie99] [DN03] [Goo04]

Process-related best practices

- Perform a feasibility study or a pilot. [RC91] [GJPD99] [Goo04]
- Implement the SMP incrementally (with a small set of software measures bottom-up, enlarge this set according to the top-down paradigm) in the manner of a high-priority and high-risk development project according to the objectives with both, a project and a risk aversion plan. [GC87] [BR88] [RC91] [Fen91] [JB93] [Pfl93] [Rub93] [GCWF95] [PGF96] [HF97] [Wie97b, Wie99] [DB99] [Kau99] [KB99] [LR99] [vSB99] [IM00] [NvV01] [Cla02] [DM02] [Den02] [Rus02] [IM03] [Jon03b] [Goo04] [IFP04] [EDBS05] [Dek05]
- State the criteria for evaluating the SMP's achievements upfront. [JB93]
- Involve all stakeholders and the developers in the SMP implementation. [RC91] [JB93] [HF97] [Kau99] [KB99] [vSB99] [GGM99] [Min00] [NvV01] [DM02] [GKMG02] [Dyb05]
- Establish a small and independent software measurement team (a user interface) of highly motivated, volunteer staff members with a huge amount of software development expertise and communication skills. [DeM82a] [GC87] [Fen91] [Das92] [JB93] [HF97] [Rus02] [Goo04] [Dek05] [Dyb05]
  1. Assign three people half-time to the team. [DeM82a]
  2. Ensure that the other half of their time is spent on something entirely different from the projects the team will be measuring. [DeM82a]
  3. Assign clear and distinct implementation and data collection responsibility and send a signal of importance. [GC87] [Mil88] [JB93] [BDT96] [Kit96b] [Den02] [Hol02] [IFP04] [EDBS05]
  4. Have the group report to someone outside the project(s) being measured. [DeM82a]
- Provide simple, complete, and consistently documented operational definitions and process descriptions for the artifacts to be measured in a certain way. [Das92] [She94] [GCWF95] [Kit96b] [Wie97b] [GGM99, GKMG02] [DM02] [Den02] [Hol02] [Goo04] [Dek05] [GMK05]

- Document procedures for the collection and analyzes of the software measures. [GC87] [Mil88] [GCWF95] [Kit96b] [BDT96] [Goo04]
- Start with projects in trouble to demonstrated the capabilities of software measurement as a valuable tool. [Pfl93] [Wie97a]
- Sell the initial collection of software measures to the collectors. [GC87] [JB93] [Min00] [Rus02] [DN03] [UE05]
- Publish what is being measured and why to ensure transparency of the software measurement process. [JB93] [Pfl93] [HF97] [GGM99] [IM00] [Cla02] [GKMG02] [DN03] [IM03] [IFP04] [Dek05] [Dyb05] [GMK05] [UE05]
- Make the software measurement data collection sufficiently easy and unobtrusive by getting tools for automatic data collection and analysis. [GC87] [MIO87] [BR88] [Fen91] [RC91] [Das92] [JB93] [Pfl93] [Kit96b] [HF97] [DFK98] [GJPD99] [vSB99] [Kul00] [Min00] [KSPR01] [Den02] [Hol02] [DN03] [Jon03b] [UE05]
- Collect and scrub the software measurement data in real-time, ensure data integrity and consistency by procedures for data entry, validation, deletion, modification and retrieval. [MIO87] [JB93] [She94] [GCWF95] [HF97] [DB99] [Rus02] [EDBS05] [Dek05]
- Store the software measurement data in an organizational database and/or repository to allow for normalization. [GC87] [Das92] [JB93] [GJPD99] [Kul00] [Rus02] [PPS03] [Har04] [Dek05]
- Integrate software measurement with the normal software processes of the organization. [SPSB91] [JB93] [KB99] [Min00] [Rus02] [DM02] [Den02] [Hol02] [DN03] [GMK05]
- Establish a review, monitor, or other mechanism to allow for the improvement and alignment the SMP's objectives and procedures to improved software processes. [GC87] [Mil88] [SPSB91] [RC91] [JB93] [PGF96] [Kit96b] [HF97] [OJ97] [Kau99] [Min00] [Den02] [IM03] [Mun03] [PPS03] [Dek05]

Measurement product-related best practices

- Ensure that measurement data offer clarity of interpretation and obvious applicability. [Rub87] [JB93]
- Ensure that the chosen software measures are relevant and acceptable to the target community. [JB93] [Jon03b]
- Facilitate actions to be taken on the basis of the observed and promptly cleaned software measurement data to display clear benefits. [GC87] [Rub87] [Fen91] [Das92] [JB93] [Wie97b] [HF97] [KB99] [DB99] [LR99] [IM00] [Kul00] [DM02] [IM03] [Jon03b] [EDBS05]
- Analyze software measurement data only for pre-defined objectives and do not abuse the results of the SMP against individuals and/or staff members. Better criticize processes or products. [GC87] [Fen91] [JB93] [Pfl93] [Wie97b] [LR99] [Min00] [Rus02] [EDBS05] [UE05]
- Grant access onto the software measures to its collectors. [Pfl93]

- Retain anonymity of individuals even if anonymity of projects and departments is impractical. [Fen91] [Wie97b] [Min00]
- Provide capabilities for users to explain events and phenomena associated with a certain, measured project. [SPSB91] [JB93] [Cla02]
- Monitor trends that key project measures exhibit over time and do not overact at single data points. [Wie97b]
- Provide prompt feedback on software measure analyses' results. [MIO87] [BR88] [JB93] [Pfl93] [Kit96b] [HF97] [Wie97a] [GJPD99] [KB99] [Wie99] [IM00] [Jon03b] [IFP04] [Dek05]
- Provide feedback and allow debate on the quality and the relevance of the SMP. [DB99] [Wie97a] [IM00] [IM03] [Dek05]
- Publicize success stories of software measurement and encourage exchange of ideas. [GC87] [JB93] [Pfl93] [Kau99] [EDBS05] [Dek05]

Best practices for SMPs following the bottom-up paradigm
Although, the bottom-up paradigm for software measurement bears a number of advantages that qualify its usage for the initial implementation cycle of a SMP, there is only a couple of authors that share their experiences with the activities contributing to success. Two of those are Hetzel and Silver [Het93, p. 33], on whose notes the following list of best practices is mainly oriented:

- Focus on the software practitioner and the work products produced.
- Define and build in software measures as part of the engineering activity bottom-up.
- Measure the inputs, outputs, and results primitives for each work product.
- Measure perceptions — what people who produce and use the work products think.
- Measure the use and results of measurements.
- Understand that the SMP will continuously evolve and change.
- Drive toward establishing validated software measures of the processes and derive meters.
- Stimulate knowledge and prepare for the next question.
- Educate management to expect and require software measures as inputs to decisions and goal setting.

2.8.5 SMP implementation steps along the measurement paradigms

In many different fields of science stage theories have been useful for developing knowledge. For instance, in the 19th century Karl Marx formulated a sceptical theory of economic development [Mar05], in which nations and their economies pass through the five sequential stages: Primitive Culture, feudalism, capitalism, socialism, or communism. The general concept behind stage theories is the premise of a description of system elements that move through a pattern of stages over time. Based on Kuznets [Kuz65], Nolan [Nol73] provides two key characteristics of stage theories: (1)

The identification of elements as characteristics for each stage, and (2) The concept of growth over time based on an analytical relationship of processes causing the change.

From an engineering point of view as proposed by Dahlbohm et al. [DM97], there is the widely established [KB99] [Dek99] [IM00] [IM00] [Col02] [BH03a] opinion that the implementation of a SMP is to be regarded as a rational engineering endeavor usually executed in the frame of a development project. A SMP is regarded like a general engineering project having the mission to transform a set of requirements under certain conditions into a deliverable and/or product. Withal, in a project requested and authorized resources are applied to complete the product within a given time frame. In so far, the project can be seen as an instance of an organization's processes. [LVJ01] This ensures the clarity of objectives, sufficient planning for staffing and/or funding and does prevent from undesired and costly surprises.

Phase	**Tasks**
Plan/evaluate	- Definition of goals, objectives, and benefits - Establishment of sponsorship - Communication and promotion of measurement - Identification of roles and responsibilities
Analyze	- Audience analysis & target software measure identification - Definition of software measures - Definition of the data collection, analysis, and data storage approach
Implement/measure	- Education - Reporting and publishing results
Improve	- Managing with expectations - Managing with software measures

Table 2.6: Phases and tasks of the 'project-approach' to implementation of SMPs (adapted from [IFP04])

In table 2.6, taken with modifications from the International Function Point Users Group (IFPUG) [IFP04], the sequential phases and the pertinent tasks are confronted. Similar lists of phases and tasks are provided by the software measurement luminaries such as Grady and Caswell [GC87], Card and Glass [CG90], or Goodman. [Goo04]

Fuchs [Fuc95] makes a good point, when he regards the evolution in the usage of software measurement paradigms from bottom-up over a mixed approach between bottom-up/top-down up to the clean top-down paradigm as stages that "... suggest how companies can introduce software measurement in an evolutionary way taking account of the company's specific experience."

Figure 2.7 illustrates the steps of implementing software measurement processes which are typically gone through along the measurement paradigm shift. A number of literature findings back up that observation. [CC93] [DFS93] [HF94] [GCWF95] Moreover, once the described staged implementation of a SMP has resulted in a definitive top-down process of software measurement, a continuous improvement of the SMP with respect to most probably ever changing measurement goals, can take place.

In recent times, especially Powell [Pow01], who examined the situation at the UK-based Rolls-Royce company, Mohagheghi [MC04, Moh04] at Ericsson in Norway, and Prechtel et al. of the German division of DaimlerChrysler AG [PSS05] corroborate the opinion of Fuchs. The latter for instance analyzed that "most companies's repositories are not collected following the GQM paradigm" and that projects being laggards of software measurement according to the top-down paradigm must be able to relate their legacy, genuine data to goals.

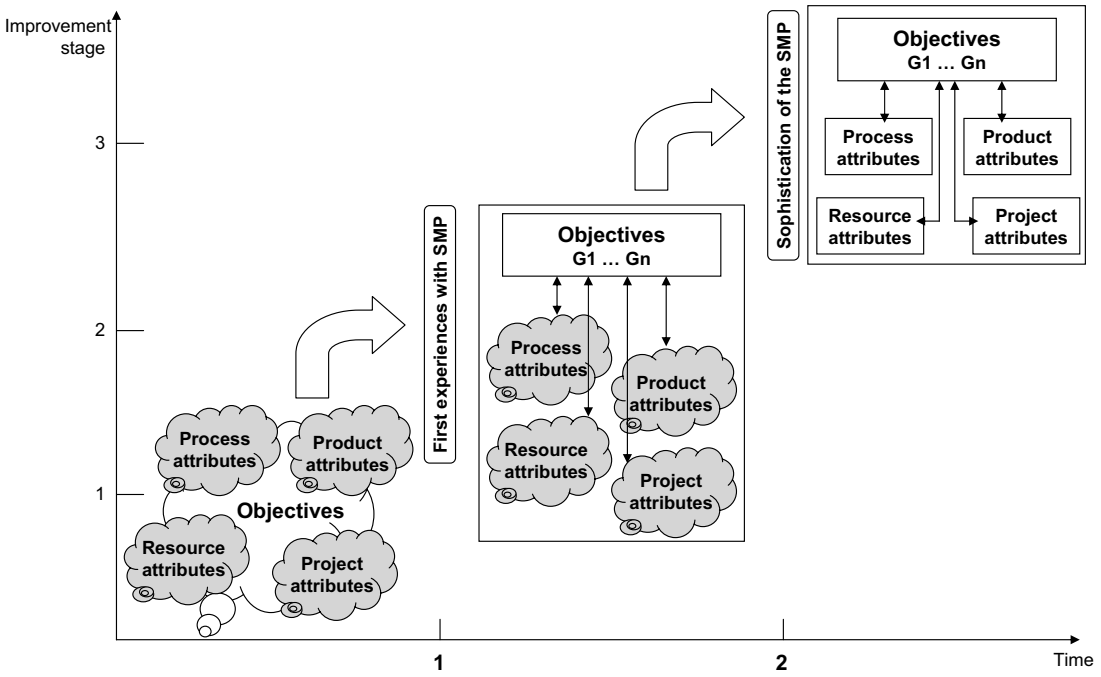

Figure 2.7: Stepwise improvement of the implementation of the software measurement process along the measurement paradigms (adapted from [Fuc95, p. 75])

2.8.6 Phases of SMP acceptance

Based on their experience, for the particular topic of SMP implementation and sustainment, Rosenberg and Hyatt [RH96b] [RH96a] [RH97] take a chance to propose an initial try to describe four stages of personnel's acceptance of SMP implementation and sustainment endeavors as depicted in figure 2.8.

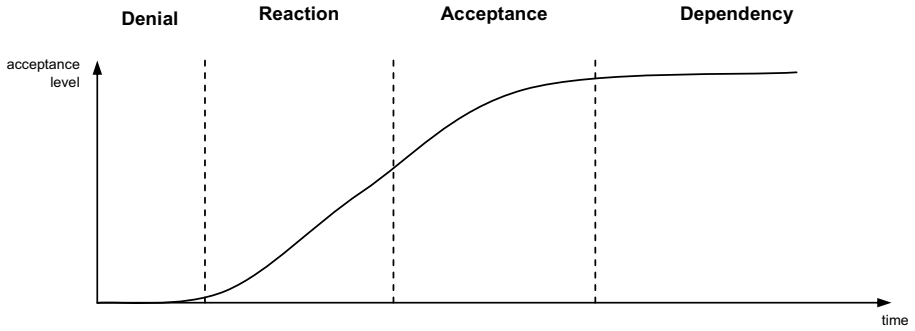

Figure 2.8: Stages of SMP acceptance among concerned personnel (adapted from [RH96a])

They suggest the following:

1. **Denial stage**
   Personnel resists to the idea in general, completely refuses to interact. Feelings of threatening can only be overcome by persuasive and social skills.

2. **Reaction stage**
   Contrary to all believes the SMP is not going to dissolve, developers must learn to live with it. Here relevant add-on value must be shown.

3. **Acceptance stage**
   Development units recognize the benefits of the SMP and incorporate software measurement into the common development structure.

4. **Dependency stage**
   The development units fully depend on software measurement in their decision-making process. Its has been adopted as cultural element.

Perceiving the term information technology (IT) in its broad sense of an artifact being based on computer communications hardware and software, Cooper and Zmud [CZ90] propose a more general, six-staged model of IT implementation activities founded on several prior models such as the famous but controversial [KK84] stage hypothesis of managing computer resources by Nolan [Nol73] or studies by Redwine and Riddle. [RR85] In doing so, Cooper and Zmud split their reflections on the process- and product-related parts:

1. **"Initiation stage**
   Process: Active and/or passive scanning of organizational problems/opportunities and IT solutions are undertaken. Pressure to change evolves from either organizational need (pull), technological innovation (push), or both.
   Product: A match is found between an IT solution and its application in the organization.

2. **Adoption stage**
   Process: Rational and political negotiations ensue to get organizational backing for implementation of the IT application.
   Product: A decision is reached to invest resources necessary to accommodate the implementation effort.

3. **Adaptation stage**
   Process: The IT application is developed, installed, and maintained. Organizational procedures are revised and developed. Organizational members are trained both in the new procedures and in the IT application.
   Product: The IT application is available for use in the organization.

4. **Acceptance stage**
   Process: Organizational members are induced to commit to IT application usage.
   Product: The IT application is employed in organizational work.

5. **Routinization stage**
   Process: Usage of the IT application is encouraged as a normal activity.
   Product: The organization's governance systems are adjusted to account for the IT application; the IT application is no longer perceived as something out of the ordinary.

6. **Infusion stage**
   Process: Increased organizational effectiveness is obtained by using the IT application in a more comprehensive and integrated manner to support higher level aspects of organizational work.
   Product: The IT application is used within the organization to fullest potential."

Despite the sequence of acceptance growth over time promoted by pertinent activities to countersteer cultural issues, Weinberg [Wei92, p. 31] claims "maturity is not the right word for subcultural patterns because it implies superiority when none can be inferred." Thus, the connection between acceptance growth and maturity is disregarded from further investigation.

2.9 Conclusion

As one part of the preparatory literature study for this thesis, the first sub-question of this research project has been addressed by this chapter:

> *RQ1. Which are general characteristics, specific cornerstones, and best practices that form content-related criteria for a potential, stepwise software measurement process improvement model?*

With respect to the theory as presented in the appendix, chapter A, the terminology of software measurement was clarified to ensure an unified vantage point for this research, in the beginning. Then, the classes of entities and possible attributes of interest for software measurement in industrial settings were presented. Afterwards, the importance of software measurement was investigated describing intentional and negative aspects, the aspired value as well as the respectively concerned audiences. When it came to paradigms applicable to software measurement, it was described, that beyond the multiply praised top-down paradigm, the bottom-up paradigm is often preferred as a good starting point. Furthermore, a mix between both can be a meaningful intermediate step, as well. Although the top-down paradigm, that is, the goal-driven software measurement process is the only commonly recommended one in literature, a subsection of this chapter threw light on the discussion of the category of top-down process models and a single, but invaluable mixed software measurement process model. The final section of this chapter entirely dealt with the topic of software measurement programs. After pointing on the situation of those initiatives in industry, costs and benefits were confronted. After all, typical implementation and sustainment pitfalls, but also an inventory of related best practices were compiled. The chapter was rounded off by exposing the typical steps of software measurement process implementation and sustainment along the software measurement paradigms.

What remains as the most important quintessence from this chapter with respect to a step-wise improvement model of software measurement process implementation as answer to the fundamental research question and sub-question number one, is reflected by the following model-related evaluation criteria.

Content-related evaluation criteria
Because it is the dedicated task of this research project to improve the implementation of software measurement processes in industrial settings, the scope of any current process improvement models to be assessed or of a one to be newly developed, must be on that aspect, as well.

> *C1. The **scope** of the process improvement model must be (at least partially) on software measurement process implementation in industrial settings.*

Already early opinions of i. e. El-Emam et al. [EMM93] see a major criteria, potential process improvement models should meet irrespective of their scope: They claim that those models should reflect sophisticated research with scientific rigor and significant merit to the community. Among others, especially the relevant measurement theory as described by the author in the appendix, chapter A must be observed in order to give evidence of sophisticated research in the area of software measurement. This leads to the following generic criterion, which requires such a model to be free from obvious evidence of design flaws:

> *C2. The process improvement model must have been developed with **scientific rigor**.*

After all, the result of the examination of industrial SMPs as described in the chapter at hand forms the last content-related evaluation criteria:

> *C3. The process improvement model must be able to reflect the sequential application of the 'bottom-up', 'mixed', and 'top-down' **measurement paradigms** during implementation of software measurement processes in industrial settings.*

Ultimately, the lesson learnt from this chapter is that the view of the process model of ISO/IEC Standard 15939, that is, one being merely aligned with the top-down paradigm, might require to be complemented with the other paradigms.

Chapter 3

Software process assessment and improvement models

"It's not enough that we do our best;
sometimes we have to do what's required."

*– Sir Winston L. S. Churchill * –*

3.1 Introduction

When the American statistician W. Edwards Deming introduced his closed-loop quality improvement approach, later slightly modified and denominated as Total Quality Management (TQM) by the U. S. Naval Air Systems Command [KBS94], to the hit rock bottom Japanese industry after World War II, there was probably no portent of how helpful his ideas could become for quality improvement of software products. [Ish85] [Zul01] Teaching the Japanese how to combine highest quality demands with fast and cheap production of goods and/or deliverance of services Deming laid the foundations for the initial and ongoing economic success of Japanese industry [Ima86] [SG02] with the Toyota Motor Company leading the way. [JMPW93] Beyond cultural changes focusing on the empowerment of quality management staff, encouragement of their questioning attitude, and creating an industrial climate that promotes a fruitful exchange of ideas [Geo03], one of his promising premises was the general concept of SPC: It urges management to disengage intervening low performance — for the time being — and to get people and processes whose performance is out of (statistical) control, that is unpredictable, under control. An improvement of the low performance of people or processes is feasible as soon as predictability is ensured. [Dem82b, Dem86] Other luminaries of the field of quality such as Feigenbaum [Fei61], Crosby [Cro79, Cro96] or Juran [Jur03, Jur06] confirm this opinion.

In the second half of the last century the gold rush fever of the 'computing age' [Gla94] did stumble about the already mentioned *software crisis* [Roy91] caused by software development projects that constantly exceeded their allotted budget and/or schedule and yielded to software products not being competitive because of their disastrous quality levels. Painfully the software engineering industry had to discover that permanently correcting bad software is inefficient and tantamount to "scraping burnt toast". [Dem86] Because the successes of solely trying to solve the problem by inserting technology were modest [VCW+84], three methodology-based approaches to tackle the issues causing the software crisis evolved.

*Prime Minister of the United Kingdom of Great Britain and Northern Ireland during World War II and winner of the 1953 Nobel Prize in literature, *1874 – †1965

First, emphasis was placed on Deming's idea of quality improvement by predominantly improving processes, once they could be brought into a state of statistical control. Despite Deming's approach being intrinsically intended for manufacturing industries, it seemed applicable to all industries, in principle. Tracing back to the psychologist Maslow [Mas43] and having been seized by Likert [Lik67] and at first for the IT sector by Nolan [Nol73] in his *staged hypothesis* of computer resources, Crosby [Cro79] [Cro96] developed the idea of a Quality Management Maturity Grid (QMMG). Applying these concepts together, practitioners such as Radice [RROC85, RHMP85] treated the task of developing a software product as a process, which can be defined, implemented, measured, controlled, and thus stage-wise improved. Later on, having the mission to develop a contractor assessment methodology for the DoD Humphrey [HS87, Hum88, Hum89] evolved this approach to a maturity framework for software processes. Resulting in elevated significance in research and important guidance for senior management in practice, SPA and its natural consequence SPI became the means of choice not merely for software quality improvement and software project control, but also for endeavors to increase productivity. [EMM93] Second, based on a cognition of the software development task different from a conceptualized process and involving a certain amount of human heroism needed to take initiative for solving ambiguous problems [Bac95] [Jak00], the Peopleware-approach of DeMarco et al. [DL99] was proposed. By regarding software mainly as a product of the developers and project managers that solve a problem under usage of processes in place, the focus was on improving people-related issues. [Wei92] Third and last, there was and still is the "Cowboy" or "Big magic" approach to software development, where a single individual, the pathological hero, can create extraordinary superb software with the aid of magic and with no other obvious support.

Probably due to the economic power of the DoD, which favored the process definition and control approach, and because it was probably in large shares the pathological heroism approach, which caused the software business to make a false step, the notion of SPA/SPI could finally succeed. [Bac95] Today, so-called capability maturity models represent specific subject-matter best-practice guidelines and assessment methods altogether. Similar to the task of developing software, the supportive sub-process of software measurement can also be regarded as a process which can be defined, measured itself, controlled, and also improved when under control. Thus, the characteristics of current software engineering capability maturity models as presented in the appendix, chapter B that also have been manifested in the ISO/IEC Standard 15504, might represent the yardstick for the process improvement models that have been proposed for software measurement implementation.

3.2 Basics of software process engineering

For scholars with a background of conventional computer science, the term 'process' will certainly imply the execution of a software program's subroutine. [Han73] Contrary, within the discipline of software engineering, processes may have an additional meaning that is strongly influenced by management sciences. [JMPW93] With the intention to provide an agreed-upon terminology around the concepts of software processes to enable others to understand and build on it, beyond others like Lonchamp [Lon93] the process pioneers Feiler and Humphrey [FH92, FH93] have gained an undisputable, major stake in the apparent triumph of software process engineering as dedicated field of software engineering. [Kin99] Beyond general terms, they structured the description of concepts in frameworks relative to the logical sequence of defining, engineering, and enacting a process. Moreover, static and dynamic process

properties are described and domain-specific interpretations are given, too. In general, their pioneering work is thorough and felicitous and therefore often referred. [Apr05]

But because research and practice have not persisted on their viewpoints and concepts, e. g. Wang et al. [WKDW99, WK00] prepared a more recent and unified framework and terminology of process system standards and models in software engineering.

3.2.1 Software process modeling

For software process engineering, where the development of software is set in analogy to a conventional manufacturing activity, two problems exist: On the one hand, the software product is intangible and invisible thereby hindering the entire development activity. On the other hand, in the context of more and more complex software products, detailed and understood process descriptions are required that can be easily instantiated to guide software engineers through imponderableness. [Ost87] While the first issue has to be accepted, process descriptions as part of a process system modeling methodology [Mad91, MS91] aid in resolving the latter issue and make the software process explicit as prerequisite for SPA/SPI. Wang et al. [WKDW99] summarize:

> **Definition 3.1:** "A *process model* is a model of a process system for describing process organization, categorization, hierarchy, interrelationship, and tailorability."

Withal, process models "... are used to help people explore the possible consequences of actions before they take them;" and "... for routine decision support where the models form an essential and automatic part of the management and control of an organization." [Pid99] [BH03b] A simplified visualization of the software process is provided in figure 3.1. [DBB+06b]

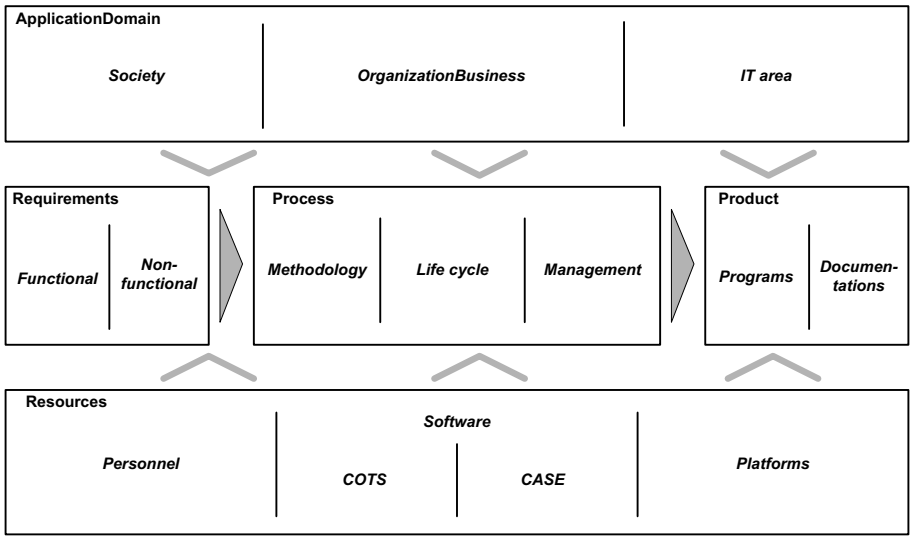

Figure 3.1: A simplified visualization of the software process

Process model categories

From the historical perspective, relative to the intended rigor of the respective process models, Madhavji [Mad91] or Lonchamp [Lon93] distinguish between *descriptive* process models [Dow86] focusing on the description of existing ways for reasons of expressing or analyzing and *defined and/or prescriptive* process models [Zav86] focusing on the desired ways of producing software for reasons of guiding and enforcing. Scarcely established, there are *proscriptive* process models [Hei90], as well, that shall

prohibit false programmer actions. However, the features of a process are mainly related to the structure. The more recent view of Wang et al. [WK00, p. 40] is reflected by the following categorization:

> **Definition 3.2:** "An *empirical process model* is a model that defines an organized and benchmarked software process system and best practices captured and elicited from the software industry."

> Because there is a number of empirical process models, a note on benchmarking a software process system shall be allowed: In chapter 2 equation 2.7 ($SDP = (M_{SDP}, R_{SDP}) = (M_{SD} \cup M_{SP} \cup M_{SR}, R_{SDP})$) was used to describe a software development project. In that context, *benchmarking* should be understood as compiling and highlighting (predominantly structural) promising software development project features from in-house or external investigations. Most often, the advantages of these project features have been proven quantitatively in terms of measurement information products (I):
>
> $$r_{SDP}^{benchmarking} \in R_{SDP} : M_{SD} \times M_{SP} \times M_{SR} \longrightarrow I$$

> **Definition 3.3:** "A *formal process model* is a model that describes the structure and methodology of a software process system with an algorithmic approach or by an abstractive process description language."

> **Definition 3.4:** "A *descriptive process model* is a model that describes 'what to do' according to a certain software process system."

> **Definition 3.5:** "A *prescriptive process model* is a model that describes 'how to do' according to a certain software process system."

Process system taxonomy
Moreover, Wang et al. [WK00, p. 52] provide a taxonomy of elements generally applicable to all process system modeling. From a top-down view a process system consists of process subsystems, process categories, processes, and practices:

> **Definition 3.6:** "A *practice* is an activity or a state in a software engineering process which carries out a specific task of the process."

> **Definition 3.7:** "A *process* is a set of sequential practices, which are functionally coherent and reusable for software project organization, implementation, and management." [similar to the definition 2.6 of a software engineering process]

> **Definition 3.8:** "A *process category* is a set of processes that are functionally coherent and reusable in an aspect of software engineering."

> **Definition 3.9:** "A *process subsystem* is a set of process categories that are functionally coherent and reusable in a main part of software engineering."

> **Definition 3.10:** "A *process system* is an entire set of structured software processes described by a process model."

While the complete terminological framework for software process modeling can be found elsewhere [FH92, FH93] [Lon93] [WKDW99, WK00], based on Curtis et al. [CKO92] there are three more basic terms to be defined:

> **Definition 3.11:** "A *process agent* is an actor (human or machine) who performs a process element."

> **Definition 3.12:** "A *process role* is a coherent set of process elements to be assigned to an agent as a unit of functional responsibility."

> **Definition 3.13:** "A *process artifact* is a product created or modified by the enactment of a process element."

Process modeling perspectives
According to Curtis et al. [CKO92] meaningful process modeling should encompass the following four perspectives:

- The *functional perspective* allows to recognize, 'what elements' of the process system taxonomy are performed and 'what information entities' are affected by the flow.
- The *behavioral perspective* makes clear, 'when' and 'how' the elements of the process system taxonomy have to be performed.
- The *organizational perspective* clarifies 'where' and 'by whom' the elements of the process system taxonomy have to be performed and what physical communication is required.
- The *informational perspective* points out, 'which information entities' are produced or manipulated by the performed elements of the process system taxonomy.

Process modeling domains
The "set of functional coverage that a process model specifies at different levels of the process taxonomy" [WKDW99] is reflected by the domain of a process model. Therefore, Wang et al. distinguish between processes applicable to the entire software developing organization (*organizational domain*), to technical software producers (*development domain*), and to management of software development (*management domain*).

Process Modeling Language (PML) types
Over the time numerous PMLs have been proposed and developed. While authors such as Curtis et al. [CKO92] provide classifications that received rather minor attention, Ambriola et al. [ACF97] prepared the probably best known classification of PMLs. [Zam01] They distinguish three different kinds according to their usefulness in process modeling:

- Process Specification Languages (PSLs) supporting requirement specification and assessment.
- Process Design Languages (PDLs) offering features of importance for the design phase.
- Process Implementation Languages (PILs) being mainly used for implementation and monitoring.

A list of the assignment of existing PMLs to the categories is provided by Ambriola et al. ibidem.

3.2.2 Software process establishment

Organizations willing to jump on the bandwagon of process-based software engineering basically follow a sequence of steps at organizational and process level: Initially, an appropriate process system reference model has to be selected and then tailored by the organization. Then, a process model for the project level has to be derived and possibly extended. After all, the resulting model should by applied and/or adapted in every-day's software development practice. Hence, the activities of tailoring, extending, and adapting to the organization's structure and culture are of essential importance. Accordingly, Wang et al. [WK00, p. 41] define:

> **Definition 3.14:** "Software process establishment (SPE) is a systematic procedure to select and implement a process system by model tailoring, extension, and/or adaption techniques."

A simplified visualization of starting software process establishment is given in figure 3.2. [DBB+06b]

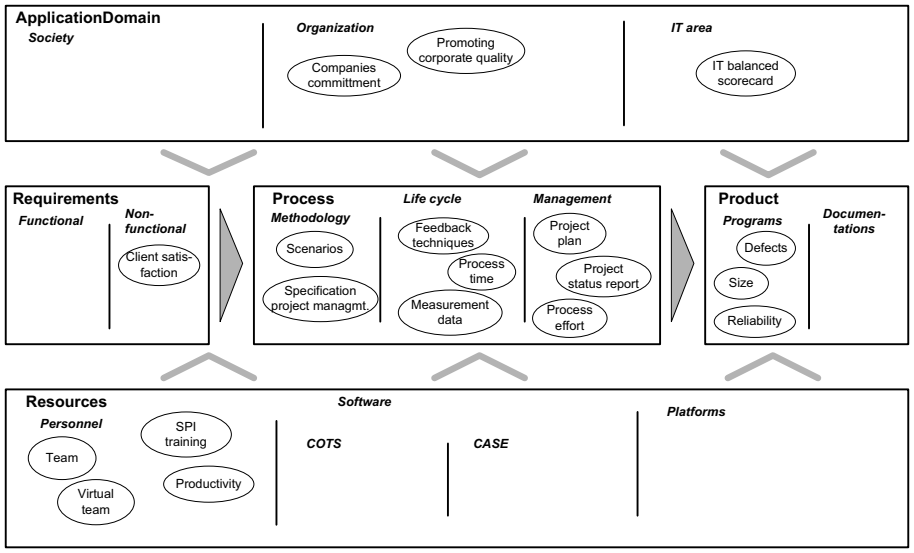

Figure 3.2: A simplified visualization of software process establishment

Wang et al. [WK00, p. 103] further specify the activities:

> **Definition 3.15:** "*Process model tailoring* is a model customization method for making a process model suitable for a specific software project by deleting unnecessary processes."

Definition 3.16: "*Process model extension* is a model customization method for making a process model suitable for a specific software project by adding additional processes."

Definition 3.17: "*Process model adaptation* is a model customization method for making a process model suitable for a specific software project by modifying, updating, and fine-tuning related processes."

3.2.3 Software process assessment

Having done excellent work in producing an encyclopedic repository of terms and concepts extracted from prevalent quality improvement initiatives in the area of software engineering, Ibrahim et al. [IH95] published a related technical report which clarifies basic terminology of importance to SPA. They summarize their findings and postulate that a characterization of processes should span on aspects of capability, performance, and maturity, which Paulk et al. [PCCW93b] describe as following:

Definition 3.18: "*Software process maturity* is the extent to which a specific process is explicitly defined, managed, measured, controlled and effective."

Definition 3.19: "*Software process capability* describes the range of expected results that can be achieved by following a (software) process."

Definition 3.20: "*Software process performance* represents the actual results achieved by following a (software) process."

Especially Card [Car91] and later Florac et al. [FC99] alert to strictly distinguish between process capability and process maturity. Accordingly, he states that initial efforts are required to establish maturity by gaining statistical control of the process, then minimizing variation around an expected level of process performance. Not earlier than then, process capability, to which other disciplines refer as a quantitative level of performance in terms of special rates, can be improved. Hence, process maturity enables the improvement of process capability as a cornerstone for increased process performance.

Ultimately, providing 'ripeness' connected with the notion of evolution or ageing as the literal meaning of the term 'maturity', Fraser et al. [FMG02] engross the thoughts.

Software process assessment vs. process maturity determination

Apparently, a general misunderstanding poaches among the contemporary community concerned with SPA/SPI expressing the fear that current software process improvement methods could just serve as instruments for an assessment-based rating, and hence a comparison, of competing organizations. [Jar00] Quite the contrary holds: Software process assessments are the logical preconditions for and thus typically come together with software process improvement initiatives that shall make software development organizations more competitive. [Zah98] [She01] They are basically conducted to gain situational process evaluations for baselining (rating) the status quo, planning improvements on weak process areas, and monitoring the improvement progress in cycles.

While so called *audits* of the software process aid in checking the compliance with a certain standard [KT04], under the umbrella of the *assessments* of the software process different types and modes occur, for which e. g. Jarvinen [Jar00] provides the

attempt of a typology. Wang et al. [WK00, pp. 42] proceed further and make strong differences between two distinct but chronologically dependent steps: The assessment of the software process and the subsequent determination of the software process' capability based on the assessment results. But attention, this statement has to be taken with a pinch of salt! Wang et al. obviously ignore the existing relations between process maturity, capability, and performance as explained by Card [Car91] when they delusively speak of 'software process capability determination' instead of the correct term 'software process maturity determination'. To circumvent any fallacy, the latter denomination will be used for the context of this thesis. Any other adopted definition of Wang et al. exhibiting this weakness in terminology is corrected in the same manner and earmarked, accordingly.
After all, the only slightly modified definitions of Wang et al. [WK00, p. 42] are as following:

> **Definition 3.21:** "*Software process assessment* SPA is a systematic procedure to investigate the existence, adequacy, and performance of an implemented process system against a model, standard, or benchmark."

> **Definition 3.22:** "*Process maturity determination* is a systematic procedure to derive a maturity level for a process, project, and/or an organization based on the evidence of existence, adequacy, and performance of the required practices defined in a software engineering process system." ['Capability' replaced by 'maturity'!]

Process assessment model
So, when tying all together, SPA refers to the first step of the on-site activity of assessing the performance of an organization's implemented software engineering process system and the process maturity determination is the subsequent step. In the literature as per Wang et al. [WK00, p. 54] both steps are supported by a *process assessment model* that consists of a *process maturity model* and a *process maturity determination method*:

> **Definition 3.23:** "A *process maturity model* is a measurement scale of software process maturity for quantitatively evaluating the existence, effectiveness, and compatibility of a process." ['Capability' replaced by 'maturity'!]

> **Definition 3.24:** "A *process maturity determination method* is an operational model that specifies how to apply the process maturity scales to measure a given process system by a process model." ['Capability' replaced by 'maturity'!]

For the process maturity model, these auxiliary means are commonly manifested:

- Practice performance scale (Measurement in terms of confidential degrees for the existence, adequacy, and effectiveness of the activities of the software process.)
- Process maturity scale (The set of process maturity levels.)
- Process maturity scope (The restriction of the assessment's statements to a single practice, a process, a project, or the entire organization.)

Believing in the notes of Greenstein [Gre05], maturity in IT and in software engineering is a concept which is applicable from different point of views. One aspect of maturation is that of the *technical equipment*, where "technical maturity describes the rate and direction of technical progress along an experimental frontier." Another aspect is *market maturity* expressing that "markets mature when returning users, instead of new users, dominate demand." After all, highly valued *organizational maturity:* occurs, when "mature organizations embody a set of routines and processes that reflect management principles for describing how the company will deliver value. The primary benefit of organizational maturity is strategic consistency."

Ultimately, Paulk et al. [Pau93b] define a process maturity level as following:

> *Process maturity level: "*A process maturity level is a well-defined evolutionary plateau toward achieving a mature software process."

In professional parlance, the term 'capability maturity model' is the more often used denomination for both, process assessment models and process improvement models as they are introduced in the next section. For instance, Aaen [Aae03] defines:

> *Capability maturity model: "*A capability maturity model describes the stages software organizations go through as they define, implement, measure, control, and improve their processes."

3.2.4 Software process improvement

As a natural consequence, the results of SPAs should be fed into improvement action plans [FR89] for weak process areas that promise tremendous return on investments (ROIs). Despite dichotomies being present in ROI's definition and having general difficulties to attribute these ROIs to those initiatives [BJ95], ratios of up to 19:1, with an average of 7:1 are often stated. [Dio93] [HCR+94] [Kra01] [Ric04] [vS04] In general, this attempt is called SPI and defined by Wang et al. as following [WK00, p. 42]:

> **Definition 3.25:** "*Software process improvement* SPI is a systematic procedure to improve the performance of an existing process system by changing the current processes or updating new processes in order to correct or avoid problems identified in the old system by means of a process assessment."

Note that in SPI it holds that software measurement goals are identical with improvement goals ($G = G_{improvement}$) and must be specified in the IT area depending on the concrete process, product, and resource, situation of a software development project.

Apart from brute force, nowadays, there are at least two basic approaches to SPI [Car91] that can be either conducted either in a bottom-up [Zuc95] or top-down [Jak98] manner: *Analysis* and *benchmarking*. Moreover, Mutafelija and Stromberg [MS03] provide a more detailed description of improvement approaches. However, because Card's argumentation that all standards and models have been derived from benchmarking in industry can be comprehended more, his categorization is used.

On the one hand, the analysis approach, or 'inductive approach' as Briand et al. [BEM95a] call it, is mainly based on the situational assessment in terms of quantitative data that can be analyzed to identify areas where corrective action is required. It is today for example manifested in the PDSA cycle of Shewhart [She31], its variations like Deming's 'Plan-Do-Check-Act' (PDCA) [Dem86], or as the underlying principle of the evolutionary Quality Improvement Paradigm (QIP) concept for learning and improvement of Basili et al. [BG94] [BC95] [Sta02]

Alternatively, there are more structured approaches and/or methodologies that aid in conducting meaningful SPI that Wang et al. [WK00] p. 61 define as:

> **Definition 3.26:** *"*A *process improvement model* is an operational model that provides guidance for improving a process system's capability by changing, updating, or enhancing existing processes based on the findings provided in a process assessment."

A simplified visualization of the relationships in a software process improvement model is provided in figure 3.3. [DBB+06b]

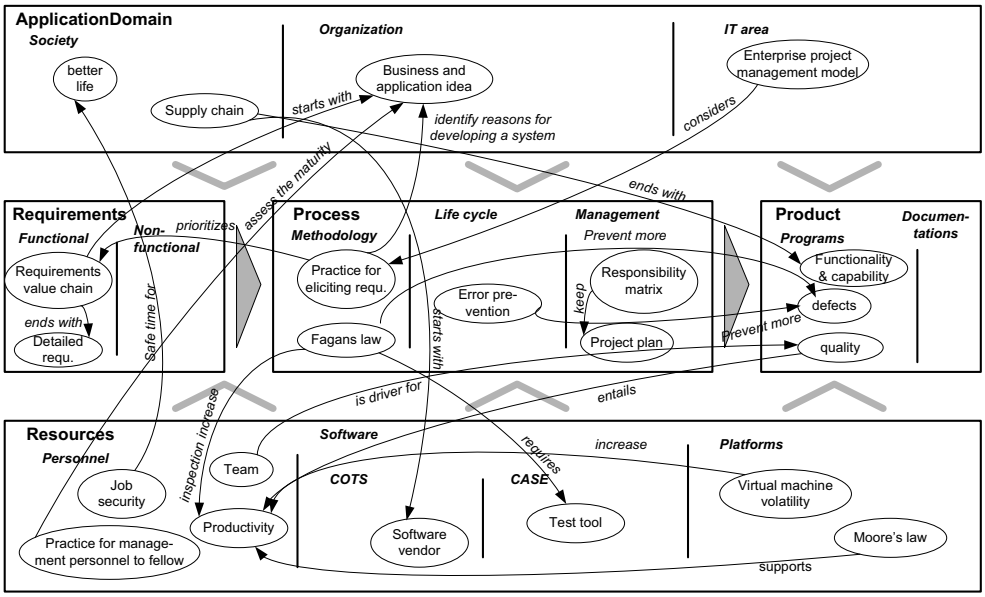

Figure 3.3: A simplified visualization of the relationships in a software process improvement model

The Benchmarking approach tries to identify organizations that outperform an industrial sector's performance. Resting on the assumption, that activities and processes of the extracted organization ensure excellence, they are proposed to be adopted. The probably most renowned and first representative of the benchmarking approach is the SW-CMM of the SEI together with its process appraisal methods and the improvement methodology Initiation – Diagnosis – Establishment – Acting – Leveraging (IDEAL). [McF96] However, by copying superior processes from the leading organization as extracted by benchmarking activities without understanding background principles and tailoring them to the own context, a company seems to be condemned to be an imitator lacking of knowledge for autonomous improvement and innovation. Thus, Deming was strictly opposed to the SW-CMM because by copying processes organizations can only catch up close, but not surpass leaders. [Zul01] Another obstacle of the benchmarking approach turns out to be the abstracting away from what is running bad.

From the vantage point of presence, the analytic/inductive and benchmarking approaches have to be necessarily combined in order to use their advantages to full capacity and retain a minimum of obstacles. [Jar00] [WKDW99] This procedure is for instance acknowledged in the criteria for the Malcom Baldridge National Quality Award (MBNQA)* of the U.S. National Institute of Standards and Technology (NIST). [Kan95]

*http://www.quality.nist.gov/

3.2.5 Software process standardization

Almost all standardization attempts in the area of software engineering aim at putting on record and optimizing existing best practices of software development either proposed by research or successfully applied in industry over years. [KS04] Apart from semi-governmental organizations such as the DoD or the European Space Agency (ESA) that urge their subcontractors to observe their standards and to use the proposed vocabulary, there are other important standardization bodies in software engineering such as the ISO, the IEC, the IEEE, or the ACM. [WK00] The probably most renowned standards in the field of SPA/SPI are ISO/IEC Standards 9000:2000, 9001:2000, 15504 (SPICE) [†], and SWEBOK [‡] [AMBD04] of IEEE.

3.3 SPA/SPI under the terms of ISO/IEC Standard 15504

After some two years of preparatory work, there was consensus among international experts that the software engineering community was in desperate need for an international standard unifying and harmonizing the myriad of different approaches to SPA/SPI. [DB98] Thus the deliberately created Working Group 10 of ISO/IEC Joint Technical Committee #1/Sub-Committee 7 (JTC1/SC7) approved a special project called SPICE in 1993 [Dor93], that should pave the way for fast development and expert involvement. Having in mind the *vision of a repeatable, comparable, and verifiable assessment method for (software) processes* that can be also beneficial for SPI , the SPICE project tried to integrate experiences from mainstream SPA/SPI models such as the Capability Maturity Model (CMM), TRILLIUM, BOOTSTRAP, and the like. [Dro95, DB98] After the release of ISO/IEC Technical Report 2 15504:1998 [EDM98] indicating a future opportunity for an affirmation, the following five parts have step-by-step become approved and standardization of the entire ISO/IEC Standard 15504 *Information Technology — Process Assessment* has been completed in the year 2006:

- ISO/IEC 15504-1:2004 — Concepts and vocabulary (*normative*)
- ISO/IEC 15504-2:2003 — Performing an assessment (*normative*)
- ISO/IEC 15504-3:2004 — Guidance on performing an assessment
- ISO/IEC 15504-4:2004 — Guidance on use for process improvement and process capability determination
- ISO/IEC 15504-5:2006 — An exemplar process assessment model

3.3.1 SPA-related regulations

In the strict sense, the normative part one of the standard defines a reference model for externally or internally performed SPA which is given in figure 3.4. On the base of a Process Reference Model (PRM) and the standard's unique Measurement Framework (MF), a Process Assessment Model (PAM) can be instantiated, which is applied during the assessment process using the inputs by responsible personnel to produce assessment output. A brief overview of particularities of the model's constituents is given beneath.

[†] Standards with costs available from ISO/IEC under http://www.iso.org/
[‡] Available for free under http://www.swebok.org/

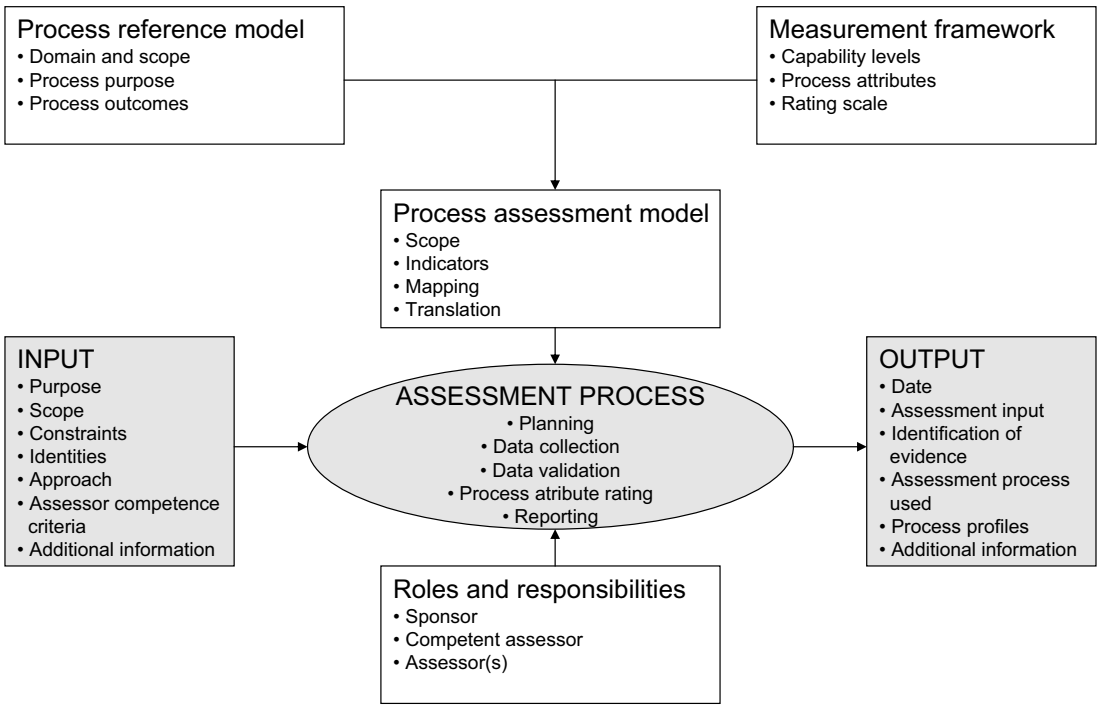

Figure 3.4: Reference model underlying ISO/IEC Standard 15504-2:2003 (adapted from the standard)

Process Reference Model
In order to ensure universal validity and optional extensibility, the standard pursuits the concept of rather relegating to PRMs, so called *plug-ins*, than providing a most probably ever incomplete process model by itself. [RS98]

ISO/IEC 15504-2 manifests the following requirements, a compliant PRM has to meet:

1. **Contents:** Declaration of the domain, description of the processes, description of the relationship between the process reference model and its context of use, and description of the relationship between the processes;

2. **General constraints:** Community consensus, uniqueness of definition and identification, limitation on normative content;

3. **Process description:** Description in terms of purpose and outcomes, set of process outcomes shall be necessary and sufficient to achieve the purpose, process description must not contain or imply parts of the measurement framework, outcome statement (production of an artifact, significant changes of state, meeting of specified constraints).

The list of available PRMs compliant with ISO/IEC Standard 15504 is already remarkable and growing [CVS+02] [KT04]:

- ISO/IEC 12207:1995/Amendment 2:2004 – Information technology – Software lifecycle processes
- ISO/IEC 15288:2002 – Systems engineering – System lifecycle processes
- ISO/IEC Technical Report 18529:2000 – Ergonomics – Ergonomics of human-system interaction – Human-centred lifecycle process descriptions

- OOSPICE Project – Component-based development processes
- SPICE-9000 for Space (S9kS) – Lifecycle processes for space-related software
- Automotive SPICE – Lifecycle processes for automotive embedded software
- Medi SPICE – Lifecycle processes for medical device software

Measurement Framework
In order to determine a certain capability level (in ascending order: 0 – Incomplete, 1 – Performed, 2 – Managed, 3 – Established, 4 – Predictable, 5 – Optimizing) of each process of the selected PRM, a rating of nine process attributes [GND98] has to be performed. There are eight generic process attributes (Performance Management, Work Product Management, Process Definition, Process Deployment, Process Measurement, Process Control, Process Innovation, Process Optimization) that are valid for all process and process reference models. Moreover, there is one (Process Performance) that covers requirements specific to the selected process reference model. [Cra98] Then, according to a ordinal process attribute rating scale the extent of achievement of each attribute is measured relative to corresponding values as provided in table 3.1.

Values in percent	Levels of achievement
0 to 15	**N**, Not achieved
>15 to 50	**P**, Partially achieved
>50 to 85	**L**, Largely achieved
>85 to 100	**P**, Fully achieved

Table 3.1: Corresponding values of the process attribute rating scale of ISO/IEC Standard 15504-2:2003 (adapted from the standard)

This represents a peculiarity, because instead of assigning a capability level relative to a discrete value from one to five, a free percentage scale from zero to 100 % can be used. [KT04] Finally, the mean of the nine process attributes is computed to rate the capability of the process under examination in terms of a rating scheme that is given in table 3.2. The confrontation of processes (*process dimension*) with the computed capability levels (*capability dimension*) turns out to be two-dimensional. [DB98]

Process Assessment Model
For the instantiation of a customized PAM on the base of the selected PRM and the unique MF, the ISO/IEC Standard 15504 establishes four categories of requirements:

1. **Scope:** Relation to at least one process of the specified PRM, addressing and declaration of all or a continuous subset of the MF's capability levels starting at level one, declaration of the scope of the selected PRM and the extracted processes;
2. **Process assessment indicators:** Declaration of indicators for performance and capability of the PRM's selected processes for explicitly addressing purposes and outcomes and demonstrating achievement of the process attributes;
3. **Mapping of the PAM to the PRM:** Complete, clear, and unambiguous mapping of the fundamental elements of the PAM to the processes selected out of the PRM;
4. **Expression of assessment results:** Provision of a formal and verifiable mechanism for the representation of the assessment results in terms of a set of process attribute ratings for each process selected out of the PRM.

Scale	Process attributes	Rating
Level 1 – Performed	Process performance	Largely or fully
Level 2 – Managed	Process performance	Fully
	Performance management	Largely or fully
	Work product management	Largely or fully
Level 3 – Established	Process performance	Fully
	Performance management	Fully
	Work product management	Fully
	Process definition	Largely or fully
	Process deployment	Largely or fully
Level 4 – Predictable	Process performance	Fully
	Performance management	Fully
	Work product management	Fully
	Process definition	Fully
	Process deployment	Fully
	Process measurement	Largely or fully
	Process control	Largely or fully
Level 5 – Optimizing	Process performance	Fully
	Performance management	Fully
	Work product management	Fully
	Process definition	Fully
	Process deployment	Fully
	Process measurement	Fully
	Process control	Fully
	Process innovation	Largely or fully
	Process optimization	Largely or fully

Table 3.2: Capability level rating scheme of ISO/IEC Standard 15504-2:2003 (adapted from the standard)

Roles and responsibilities

By and large, there are three different roles with their distinct responsibilities to be filled in an assessment compliant with this standard [Col98]: the *sponsor*, the *competent assessor*, and multiple *assessors* as worker bees. The sponsor usually cares for the appropriateness of the competent assessor for the assessment, ensures the provision of sufficient resources and grants access rights to all required sources of information for the assessment. Being in charge of directing the entire assessment, the competent assessor sees about issues such as the observation of the standard's regulations during the assessment, qualification of downstream assessors, and the like. After all, the worker bees of an assessment are represented by multiple assessors carrying out the activities required for the assessments including the rating process procedure.

In order to ensure the comparability and traceability of the assessment's results by customers [KT04], the International ISO/IEC 15504 Assessor Certification Scheme (iNTACS)* has been brought into being to oversee the certification of assessors and regulate accreditation of certification bodies. Most notably, recently a three-level certification scheme for assessors has been produced [INT06] that distinguishes between the

*http://www.intacs.info/

entry level of *Provisional Assessors*, the intermediate level of *Competent Assessors*, and the superior level of *Principal Assessors*.

The assessment process
To meet the purpose of the assessment, it must be conducted according to a documented assessment process. This process comprises the following five phases embracing the mentioned aspects:

1. **Planning:** Assessment inputs, activities, resources, responsibilities, criteria for compliance with this standard, description of the planned assessment outputs;
2. **Data collection:** Demonstration of the data collection technique, matching between the organizational unit's processes and the elements of the PAM, assessment of each identified process, recording of objective evidences for verification of ratings;
3. **Data validation:** Confirmation of objective data collection, ensuring of evidence's sufficiency for the assessment, ensuring of consistency;
4. **Process attribute rating:** Recording of process attribute ratings as process profile, usage of the gathered indicators to support judgement, recording of the decision-making process for the judgement, ensuring of traceability between the attribute ratings and the objective evidences, and recording of their relationship;
5. **Reporting:** Communication of the assessments results and defined results to the assessment sponsor.

3.3.2 SPI-related guidelines

In one of its informative parts, more precisely in number four, the ISO/IEC Standard 15504 provides a comprehensive, eight-step model (cf. 3.5) exhibiting how to embed SPA in the SPI cycle. A very brief overview is given below:

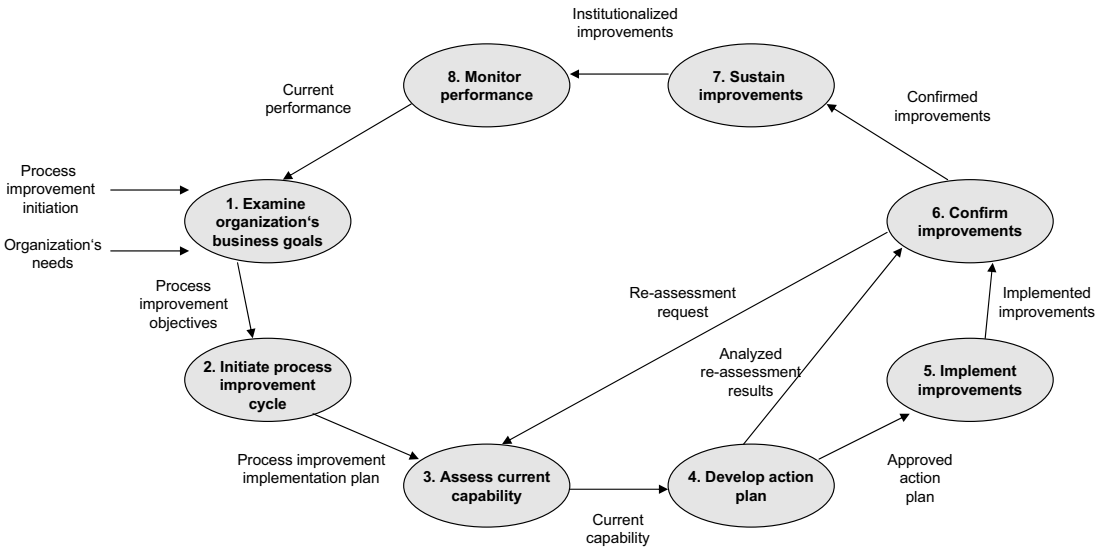

Figure 3.5: Process improvement model of ISO/IEC Standard 15504-4:2004 (adapted from the standard)

(1) Once, the idea of process improvement has been seeded, the organization's needs and business goals have to be examined to form the entire program's objectives.

(2) Then, a preparatory step is performed: The sponsor chooses the processes to be examined from the specific PRM and provides target profiles for each process in terms of relevant process' attributes and the respectively anticipated capability. Together with that information additional characteristics such as roles, responsibilities, milestones, and the like are specified in the process improvement plan. (3) Now the virtual determination of the current capabilities of the chosen processes can take place according to the instantiated PAM. (4) On the base of the determined process capabilities an action plan can be developed that defines improvement objectives and sets appropriate targets to be traced. (5) In case the action plan could gain agreement the improvements should be implemented by a project running through to typical phases of e. g. detailed planning, design, etc. (6) After completion of the implementation project(s) the organization should confirm that the improvements result in the planned objectives and that adoption and the anticipated process capability have been established. An intermediate progress check can also be performed for multi-step improvements by means of re-assessments. Because in those cases comparability is of elevated importance, the PAM insists upon stringent documentation. (7) Now, that the improvement has been confirmed, it has to be sustained at the set level of capability and possibly extended from pilot projects to the full range of an organization's projects. (8) After all, the performance of the organization's processes should be monitored in a continuous manner and new process improvements should be considered, where required. Then, the cycle starts over, again.

3.4 Conclusion

As the result of the final part of the literature study for this thesis, answers to the second sub-question of this research project have been found within this chapter:

> *RQ2. Which are the requirements for process improvement models in software engineering that form model-related criteria for a potential, stepwise software measurement process improvement model?*

This chapter started with a historical journey into the genesis of process improvement models and provided essential background knowledge of software process engineering as required in the course of this thesis. Finally, SPA regulations and SPI guidelines posed by ISO/IEC standard 15504 were exposed as the base for further investigation of the process improvement models software measurement process implementation.

Model-related evaluation criteria
To be able to evaluate existing solutions from the modeling point of view, a framework of model-related criteria is needed to prescribe the desired solution. The deduction of figures of merit from the contents of this chapter leads to these results:

*M1. The process improvement model must provide a **process reference model** compliant with the requirements of ISO/IEC Standard 15504.*

*M2. The process improvement model must be **compatible with the measurement framework** of ISO/IEC Standard 15504.*

*M3. The process improvement model's **process assessment model** must represent a mapping of the process reference model and the measurement framework of ISO/IEC Standard 15504.*

*M4. The process improvement model's **assessment process** must observe the regulations of ISO/IEC Standard 15504.*

*M5. The process improvement model must provide process **improvement guidelines** for the specified scope.*

Chapter 4

Review and evaluation of related work

"Do not go where the path may lead;
go instead where there is no path and leave a trail!"

– Ralph Waldo Emerson –*

4.1 Introduction

During the extended search for existing SPA/SPI models concerned with the scope of software measurement process implementation in industrial settings it became obvious that the importance of measurement as an engineering-relevant issue has penetrated the software engineering community for years. [JZ97] However, there are differences in the foci of the models found: Particularly, the findings fall into two categories:

- *Implicit models* do not solely focus on the maturation of the software measurement processes but require a step-by-step improvement along software engineering process maturation, on the very spot.
- *Explicit models* are those that directly deal with the structured improvement of software measurement processes.

Beneath, appreciating the previous work done by Buglione and Abran [BA03], the findings of both categories are briefly reviewed [BKFD05] [BKF+05] [Bra05] and evaluated according to the following proposed figures of merit as revealed as quintessence from chapters 2 and 3:

RQ1 (Content-related criteria)

C1. The **scope** of the process improvement model must be (at least partially) on software measurement process implementation in industrial settings.

C2. The process improvement model must have been developed with **scientific rigor**.

*U.S.-American poet, essayist and lecturer at Harvard University, Cambridge, MA, USA, *1803 – †1882

C3. The process improvement model must be able to reflect the sequential application of the 'bottom-up', 'mixed', and 'top-down' **measurement paradigms** during implementation of software measurement processes in industrial settings.

RQ2 (Model-related criteria)

M1. The process improvement model must provide a **process reference model** compliant with the requirements of ISO/IEC Standard 15504.

M2. The process improvement model must be **compatible with the measurement framework** of ISO/IEC Standard 15504.

M3. The process improvement model's **process assessment model** must represent a mapping of the process reference model and the measurement framework of ISO/IEC Standard 15504.

M4. The process improvement model's **assessment process** must observe the regulations of ISO/IEC Standard 15504.

M5. The process improvement model must provide process **improvement guidelines** for the specified scope.

4.2 Implicit models

As can be recognized from the short overview of mainstream models for SPA/SPI of software processes in the appendix, chapter B, the investigation of aspects of SMP and/or software measurement process maturation can be cut down to the models that serve as common base: CMM v1.1, ISO/IEC Standard 9001:2000, and CMMI v1.1.

4.2.1 CMM v1.1

Contents perspective

Criterion C1: Partly acknowledging the importance of measurement, the authors of the CMM and/or SW-CMM provided initial elements to assess the measurement process within software process improvement, but set aside the coherent, individual treatment of a viable software measurement process [GJR03] — not to mention processes needed for SMP implementation and sustainment. The CMM requires each Key Process Area (KPA) to be measured, analyzed, and tracked. It manifests this measurement and analysis task for the status and effectiveness of the software processes in one of the five common features ('Measurement and Analysis') applicable to all KPAs. [BA04b] Imposed by the model's structure presented in the appendix, section B.2, the accomplishment of the set of goals associated with each software process is being unearthed by collecting data for a break down of the specific goals into a series of indicators. A

first classification of indicating software measures dominant for each process maturity level has been published by Pfleeger and McGowan. [PM90] Another one, that has been advocated by the SEI, is presented by Baumert et al. [BM92] and adopted from Walrad et al. [WM93] in table 4.1.

Criterion C2: With increasing software process maturation on each distinct process maturity level, the focus of information needs changes and/or sophisticates thereby dictating a change of the focus of software measurement. [WM93] By shifting software measurement to more challenging measures [GRS98], increasing *software measurement maturity* is thought to be adumbrated. Accordingly, Weller [Wel94] of Bull Honeywell company proposed a staggered model of increased project management capability based on the increased visibility promoted by the introduction of more challenging software measures.

Moreover, an *implicit software measurement program maturity* model is arranged as the 'Quantitative Process Management' KPA on capability maturity level four, in which some basic practices are under consideration and used for compliance assessment. [WL02] Comer et al. [CC93] view the implications two-tiered: While for the specific KPA the state of conformance might inform on a SMP's maturity earliest on organizational maturity level four, measurement maturity may be deduced from the organization's maturity level. This opinion is vehemently criticized by John McGarry and colleagues [BJC+96]: They agree that measurement maturity of an organization be one specific dimension of process maturity, but observe that very often SMPs implementation and sustainment processes's maturity lacks behind the overall software process maturity level. Because the implicit SMP maturity model has been refuted by *reductio ad absurdum*, it will be disregarded for a lack of scientific rigor.

Criterion C3: In addition to the practices covered in the common features that generally contribute to SPI and software measurement endeavors, the CMM provides merely a small set of seven only partially related best practices specific to software measurement process implementation and sustainment first in the level 4 KPA 'Quantitative Process Management'. These abridged practices are listed beneath [PWG+93, p. A-57]:

- "The software project's plan for quantitative process management is developed according to a documented procedure.
- The software project's quantitative process management activities are performed in accordance with the project's quantitative process management plan.
- The strategy for the data collection and the quantitative analyses to be performed are determined based on the project's defined software process.
- The measurement data used to control the project's defined software process quantitatively are collected according to a documented procedure.
- The project's defined software process is analyzed and brought under quantitative control according to a documented procedure.
- Reports documenting the results of the software project's quantitative process management activities are prepared and distributed.
- The process capability baseline for the organization's standard software process is established and maintained according to a documented procedure."

As can be easily recognized from a simple comparison of the best practices as introduced in subsection 2.8.4 with the short list above, the statements are a few generalities that do not even mention a single measurement paradigm. Thus this criterion is not met, at all.

CMM level	Aim to advance	Dominant software measures, samples
Level 1	Achieve rudimentary schedule predictability	**Post-ship measures** - Customer-detected defects - Production failures - Application size at time of delivery - Total development cost and time
Level 2	Stable process, cost predictability	**Project control** - Trouble report density - Actual vs. planned completions/costs - Number of requirements changes - Application size growth
Level 3	Formal life cycle, methodology, tools, and technology	**Product control** - Defect detection efficiency - Defect frequency by defect type - Requirement change distribution by requirement type - Actual vs. planned completions/costs with range
Level 4	Productivity plans, goals measures, tracking; process improvement, increased defect prevention	**Process control** - Progress in relation to control limits - Cost in relation to control limits - Size growth in relation to control limits - Defect removal efficiency - Rework measures
Level 5	Continuous improvement of the process(es); measurement system evaluation and refinement	**Quality management** - Actual vs. predicted process improvement results - Reductions in process variance - actual vs. predicted process innovation effects - Innovation adoption rates - Rework effort as percentage of total effort

Table 4.1: Advocated measures dominant for each process maturity level of the CMM (adapted from [WM93, p. 455])

Model perspective

Criterion M1: Obviously, the PRM of the CMM does not comply with the specifications made by the regulations of ISO/IEC Standard 15504. Although, the basic contents and the general constraints are mentioned, the process description lacks the description of purposes and specific outcomes.

Criterion M2: Because the measurement framework of the CMM cannot be completely mapped onto the regulation set out by ISO/IEC Standard 15504, also this criterion has to be regarded as failed.

Criterion M3: Caused by the failing of criteria M1 and M2, the process assessment model is condemned to fail for criterion M3, as well.

Criterion M4: The assessment processes of the CMM, which have been manifested in the CMM-based Appraisal Method for Internal Process Improvement (CBA-IPI) and Software Capability Evaluation (SCE) follow the basic regulations of ISO/IEC Standard 15504. So this criterion can be regarded as successfully passed.

Criterion M5: With the improvement guidelines being given with the aid of a staggered structure of processes of the CMM, this criterion is not a problem.

4.2.2 ISO/IEC Standard 9001:2000

The focus of that standard is predominantly on implementing the processes required for a quality management system and deals with the assessment of compliance and continuous improvement of weak areas. [GRS98] Though measurement in general is recognized to be of elevated importance as for instance in clause eight by stressing its usefulness in enabling continuous improvement, a specific model is not provided. [Sch03] Thus, this standard is completely disregarded for further evaluation.

4.2.3 CMMI Framework v1.1

Contents perspective

Criterion C1: Compared with its predecessor models, in the CMMI Framework v1.1 software measurement has gained "considerable recognition among software engineering professionals". [BA04b] With the integration of several models (cf. section B.6) resulting in a staged and a continuous architecture of process arrangement, a general reduction of common features by one, and the related relocation of important (software) measurement activities to the fundamental, maturity level two, support process area 'Measurement and Analysis', the elevated importance can apparently be corroborated. The framework's authors make a good point when they state the intention of the process area: "The purpose of Measurement and Analysis is to develop and sustain a measurement capability that is used to support management information needs." [SEI02a, p. 476] It is assumed that the information needs act as input which has been previously identified by calling processes. [Zub01] In doing so, the process area is structured around two specific goals. [GJR03] A fine-grained overview of specific-practice-to-specific-goal relationships adapted from the CMMI documentation [SEI02a, pp. 478] is presented next. It is applicable for the CMMI – Measurement & Analysis (CMMI-MA) support process area of both representations:

"**Specific goal 1**: Align measurement and analysis activities

- *Specific practice 1.1-1*: Establish measurement objectives

  Sub-practices:

  1. Document information needs and objectives.
  2. Prioritize information needs and objectives.
  3. Document, review, and update measurement objectives.
  4. Provide feedback for refining and clarifying information needs and objectives as necessary.
  5. Maintain traceability of the measurement objectives to the identified information needs and objectives.

- *Specific practice 1.2-1*: Specify measures

  Sub-practices:

  1. Identify candidate measures based on documented measurement objectives.

2. Identify existing measures that already address the measurement objectives.
3. Specify operational definitions for the measures.
4. Prioritize, review, and update measures.

– *Specific practice 1.3-1*: Specify data collection and storage procedures

Sub-practices:

1. Identify existing sources of data that are generated from current work products, processes, or transactions.
2. Identify measures for which data are needed, but are not currently available.
3. Specify how to collect and store the data for each required measure.
4. Create data collection mechanisms and process guidance.
5. Support automatic collection of the data where appropriate and feasible.
6. Prioritize, review, and update data collection and storage procedures.
7. Update measures and measurement objectives as necessary.

– *Specific practice 1.4-1*: Specify analysis procedures

Sub-practices:

1. Specify and prioritize the analyses that will be conducted and the reports that will be prepared.
2. Select appropriate data analysis methods and tools.
3. Specify administrative procedures for analyzing the data and communicating the results.
4. Review and update the proposed content and format of the specified analyses and reports.
5. Update measures and measurement objectives as necessary.
6. Specify criteria for evaluating the utility of the analysis results, and of the conduct of the measurement and analysis activities.

Specific goal 2: Provide measurement results

– *Specific practice 2.1-1*: Collect measurement data

Sub-practices:

1. Obtain the data for base measures.
2. Generate the data for derived measures.
3. Perform data integrity checks as close to the source of the data as possible.

– *Specific practice 2.2-1*: Analyze measurement data

Sub-practices:

1. Conduct initial analyses, interpret the results, and draw preliminary conclusions.
2. Conduct additional measurement and analysis as necessary, and prepare results for presentation.
3. Review the initial results with relevant stakeholders.
4. Refine criteria for future analyses.

– *Specific practice 2.3-1*: Store data and results

Sub-practices:

1. Review the data to ensure their completeness, integrity, accuracy, and currency.
2. Make the stored contents available for use only by appropriate groups and personnel.
3. Prevent the stored information from being used inappropriately.

– *Specific practice 2.4-1*: Communicate results

Sub-practices:

1. Keep relevant stakeholders apprised of measurement results on a timely basis.
2. Assist relevant stakeholders in understanding the results."

While for both representations these specific goals can be accomplished with the specific practices and related sub-practices of the process area, the achievement of generic goals differs between the representations. There are five generic goals in ascending order (each belonging to the pertinent capability level in the continuous version), for which generic practices should be performed; the staged version does solely rely on generic goals two and three. Other guidance for good measurement practices are distributed and integrated with other process areas requiring measurement and analysis to be performed. Additionally, the common feature 'Direct Implementing' demands ongoing process measurement. According to Goldenson et al. [GJR03], a *maturation of measurement capabilities* with respect to the continuous representation in the context of performing other processes can be achieved as for any other process area: At first base practices required by specific goals should be implemented and then generic goals with the aid of generic practices accomplished. The following list reflects the generic-practice-to-generic-goal relationships relevant to the maturation of software measurement in the CMMI Framework as extracted from its documentation [SEI02a, pp. 478]:

"**Generic goal 1**: Achieve specific goals

– *Generic practice 1.1*: Perform base practices

Sub-practice:

* Perform the base practices of the measurement and analysis process to develop work products and provide services to achieve the specific goals of the process area.

Generic goal 2: Institutionalize a managed process

– *Generic practice 2.1*: Establish an organizational policy

Sub-practice:

* Establish and maintain an organizational policy for planning and performing the measurement and analysis process.

– *Generic practice 2.2*: Plan the process

Sub-practice:

* Establish and maintain the plan for performing the measurement and analysis process.

- *Generic practice 2.3*: Provide resources

  Sub-practice:

  * Provide adequate resources for performing the measurement and analysis process, developing the work products, and providing the services of the process.

- *Generic practice 2.4*: Assign responsibility

  Sub-practice:

  * Assign responsibility and authority for performing the process, developing the work products, and providing the services of the measurement and analysis process.

- *Generic practice 2.5*: Train people

  Sub-practice:

  * Train the people performing or supporting the measurement and analysis process as needed.

- *Generic practice 2.6*: Manage configurations

  Sub-practice:

  * Place designated work products of the measurement and analysis process under appropriate levels of configuration management.

- *Generic practice 2.7*: Identify and involve relevant Stakeholders

  Sub-practice:

  * Identify and involve the relevant stakeholders of the measurement and analysis process as planned.

- *Generic practice 2.8*: Monitor and control the process

  Sub-practice:

  * Monitor and control the measurement and analysis process against the plan for performing the process and take appropriate corrective action.

- *Generic practice 2.9*: Objectively evaluate adherence

  Sub-practice:

  * Objectively evaluate adherence of the measurement and analysis process against its process description, standards, and procedures, and address noncompliance.

- *Generic practice 2.10*: Review status with higher level management

  Sub-practice:

  * Review the activities, status, and results of the measurement and analysis process with higher level management and resolve issues.

Generic goal 3: Institutionalize a defined process

- *Generic practice 3.1*: Establish a defined process

  Sub-practice:

  * Establish and maintain the description of a defined measurement and analysis process.

– *Generic practice 3.2*: Collect improvement information

Sub-practice:

* Collect work products, measures, measurement results, and improvement information derived from planning and performing the measurement and analysis process to support the future use and improvement of the organizationŠs processes and process assets.

Generic goal 4: Institutionalize a quantitatively managed process

– *Generic practice 4.1*: Establish quantitative objectives for the process

Sub-practice:

* Establish and maintain quantitative objectives for the measurement and analysis process that address quality and process performance based on customer needs and business objectives.

– *Generic practice 4.2*: Stabilize subprocess performance

Sub-practice:

* Stabilize the performance of one or more sub-processes to determine the ability of the measurement and analysis process to achieve the established quantitative quality and process performance objectives.

Generic goal 5: Institutionalize an optimizing process

– *Generic practice 5.1*: Ensure continuous process improvement

Sub-practice:

* Ensure continuous improvement of the measurement and analysis process in fulfilling the relevant business objectives of the organization.

– *Generic practice 5.2*: Correct root causes of problems

Sub-practice:

* Identify and correct the root causes of defects and other problems in the measurement and analysis process."

Criterion C2: Starting from the vantage point of presence, in which no major scientific flaws disturb the usage of the CMMI in general and the CMMI-MA support process area in particular, this criterion is fully satisfied.

Criterion C3: When comparing the set of best practices for SMP implementation and sustainment as listed in section 2.8.4 with the above listed ones of CMMI, it becomes clear that it covers nearly all best practices with either specific or generic practices. However, the measurement and analysis support process area of the CMMI is slanted toward the top-down paradigm and completely ignores the bottom-up paradigm as basic starting point. However, sub-practice 2 of specific practice 1.2-1 and sub-practice 2 of specific practice 1.3-1 indicate that the mixed paradigm is indeed not exhaustively covered but at the same time not completely ignored.

Model perspective

Criterion M1 – M5: To all intents and purposes, the CMMI Framework v1.1 and the basic components of its PRM have been designed along the specifications made by ISO/IEC Standard 15504. Consequently, the decision criteria from the model perspective are completely met.

4.3 Explicit models

Apparently, software measurement has been regarded as a valuable means for software engineering for years. [Ebe99] But when looking at the works explicitly addressing the topics of the improvement of its implementation and sustainment in the course of SMPs, it seems that it has been treated stepmotherly. There have been some unassertive approaches as e. g. the one of Sage [Sag97], who pretends to have provided a measurement maturity model but solely aligns information needs and pertinent software measures into a sequence. In a similar manner Gantzer et al. [GRS98] do with their opinion of 'measurement capability' based on a cross-section of pertinent standards related to software measurement. Also the approach of Baxter [Bax98], who provides a framework for assessing the effectiveness of the software measurement process relative to the expected results, remains silent to a large extent. However, the process model of Mendonça [Men97] is not disregarded from the evaluation despite it is only intended to work as an improvement model.

In sum, the number of candidate models being explicitly geared towards the process improvement for software measurement process implementation and sustainment can be counted on the fingers of one hand and shall be briefly examined beneath.

4.3.1 Software measurement technology maturity

In a seminal paper, Daskalantonakis et al. [DBY91] proposed a way to assess the maturity of software measurement technology, which strongly emulates the assessment methodology for software development processes as initially proposed by Humphrey and Sweet [HS87] as the foundation for the SEI's CMM. Based on similar assumptions as the CMM, they define a pared-down version of a *maturity grid* (cf. appendix, subsection B.8) of themes lacking practices, for which they propose five evolutionary stages as adumbrated in table 4.2. For the purpose of assessment, the different stages of the themes have been transformed into a *Likert-like questionnaire* (also cf. the Fraser typology in the appendix, subsection B.8), in which respondents score the software measurement technology maturity according to the same procedure as in the SEI's CMM. Unfortunately, the concrete model and questionnaire is only available as sample from the paper, what hinders evaluation.

Later customization of the model: Motivated by a policy of the U. S. Air Force [Dru94] prescribing the use of software measurement for internal reviews and for the evaluation of external contractors, the affiliated Software Technology Support Center (STSC) developed the 'software metrics capability evaluation guide'. [BP95] [BP96a] Besides other materials, the guide developed by the STSC comprised descriptions of the framework and of the evaluation methodology and/or process. Withal, the framework was not created from scratch but turns out to be identical with the theme-based maturity grid (cf. table 4.2) for software measurement technology maturity as defined by Daskalantonakis et al. [DBY91] As a peculiarity in a first (and from the vantage point of today also last) step, the STSC does merely use a shrunk subset of themes from one to six with evolutionary stages from level one to three.

Contents perspective

Criterion C1: Although the model is designed on the lines of the CMM that concentrates on the assessment and improvement of the capability maturity of the software development process, the scope of this model is on software measurement technology for SMP. Because criterion C1 requires a model to be geared towards software measurement process implementation and sustainment, the criterion is satisfied.

Theme	**Level 1** (Initial)	**Level 2** (Repeatable)	**Level 3** (Defined)	**Level 4** (Managed)	**Level 5** (Optimized)
1. Formalization of the development process	- Process unpredictable - Project depends on experienced and seasoned professionals - No/poor process focus	- Repeat previously mastered tasks - Process depends on experienced people	- Process characterized and reasonably understood	- Process measured and controlled	- Optimized process - Focus on process improvement - Reward process improvements
2. Formalization of the measurement process	- Little or no formalization	- Formal procedures established - Standards exist	- Documented standards - Standards applied	- Improvement mechanisms in place - Internal standards applied	- Organization has learned and improved
3. Scope of measurement	- Done occasionally on project with experienced people, or not at all	- Done on projects with experienced people - Project estimation mechanisms exist - Project focus	- Goal/Question/Metric package development and some use - Data collection and and recording - Existence of specific automated tools - Product focus	- Metric packages being applied and managed - Problem cause analysis - Existence of integrated automated tools - Process focus	- Have learned and adapted metric packages - Problem prevention - Process optimization
4. Implementation support	- No data or database	- Per project database	- Product database - Standardized database across projects	- Process database - Common corporate database and process information	- Knowledge base - Improvement and learning data base
5. Measurement evolution	- Little or no measurement conducted	- Project measurement and management	- Product measurement - and management	- Process measurement and management	- Continuous feedback and improvement

6. Measurement support for management control	- Management not supported by measurement	- Some support by measurement - Basic control of	- Product measurement and control	- Management process measured and controlled	- Can define technology values and needs - Can check achievement
7. Project improvement	- No statistical process control - No senior management involvement and understanding	- Disciplined project and configuration management - Risk management - Subcontractors evaluated	- Dedicated process resources - Process data not retained nor analyzed properly - Subcontractors controlled	- Need discipline to track and eliminate problems - Improving technology	- Quantitative project feedback - Effective reuse
8. Product improvement	- Poor configuration management and quality assurance	- Effective (independent) quality assurance - Reviews conducted - Periodic customer interface for feedback	- Qualitative foundation for applying technology - Not many quantitative measures of problem causes	- Have some mechanisms for determining problem causes	- Can analyze problem causes and prevent them
9. Process improvement	- Hard to plan and commit - No focus on improvement	- Hard to improve the measurement process - No foundation for improvement	- Quantitative foundation for improvement - Training required	- Methods and tools tailored to needs - Quality improvement - Training assessed and improved	- Significant productivity improvement
10. Predictability	- Unpredictable results for similar work - Unstable plans and schedules	- Predictable results for similar work - Tools used for project planning	- Able to project and track product quality parameters	- Process quality and productivity projection and tracking	- High predictability due to process formalization and optimization - Can analyze problem causes and prevent them

Table 4.2: Maturity grid for software measurement maturity assessment (taken from [DBY91])

Criterion C2: The model is based on a number of assumptions Daskalantonakis et al. extracted from their own beliefs and observations. Consequently, the degree of scientific rigor is at least questionable if not absent.

Criterion C3: The maturity grid rather describes the result of certain practices distributed over different, staggered levels of software measurement maturity than provides best practices. However, it should be noted that the authors recognize the fact that the application of the top-down measurement paradigm (they propose GQM) reflects a degree of sophistication in the scope of measurement (theme 3). Unfortunately, they omit to provide information on the bottom-up measurement paradigm that mirrors the preceding step of maturation when believing the notes of Hetzel and Silver. [Het93] Taken altogether, this criterion is not satisfied.

Model perspective

Criterion M1 – M4: Connected with the absence of best practices caused by the pared-down version of a maturity grid, as a logical consequence the model-related criteria are not satisfied, at all.

Criterion M5: Since the model obviously does not provide a process model with best practices or any improvement guidelines other than the desired results for each theme embodied in questions of a Likert-like questionnaire, this criterion is not satisfied.

4.3.2 The measurement maturity model

In 1993, Comer and Chard [CC93] published initial results of a research project aimed at the development of a methodology to assess the software measurement process as logical precondition for an improvement approach, called the *measurement maturity model*. Having the premise to build a process model, in which the software measurement process can be reduced to a sequence of key processes that can be assessed individually, the researchers dismiss the idea of prescribing certain software measures to be collected as previous researchers did. Fortunately, Comer and Chard recognize at least the top-down and bottom-up types of measurement paradigms and propose their coverage in the model. Moreover, they also recognize that the "maturity of the measurement programme is directly related to the ability of the organization to define adequate goals." [CC93, p. 288] The key processes of software measurement and a working excerpt of activities of the key processes as identified during their research project are:

1. Process definition (specification of entities and attributes, identification of goals, derivation of software measures);
2. Collection (operational definition of software measures, data gathering, data storage, verification, examination for patterns);
3. Analysis; and
4. Exploitation.

The approach to the development of an assessment methodology is characterized by the pursuit of compatibility with the one of the SEI's CMM as proposed by Humphrey and Sweet [HS87] using the transformation of answers to a questionnaire to a grading system. In doing so, weighted questions check the conduction of activities, the laying of general organizational foundations to support the activities, and the effectiveness of the activities. However, the researchers do explicitly state their intention to highlight improvement areas rather than grading an organization on a scale of software measurement maturity.

With the model exhibiting a considerable share of the desired attributes, it is a pity that the mentioned working paper remained the only proof of life leaving an incomplete and only adumbrated model for the public. Because there is not enough material to review and evaluate further this model, it is disclosed from further recognition.

4.3.3 The META Measurement Maturity Model (4M)

In contrast to what the title threatens, the marketing-based term 'META Measurement Maturity Model (4M)' does not keep its promise when believing in the appearance of Malcolm Slovin's article. [Slo97] It rather represents a minimum maturity grid that is (for reasons of better marketing through a recognition factor) loosely coupled with elements similar to those of the SEI's CMM. The five-level maturity grid along more extensive and META-group consulted software measurement can be summarized as:

1. *Level 1 — Chaotic*
   Unpredictable costs, schedules, quality, and performance.
2. *Level 2 — Intuitive*
   Costs and quality are variable while schedules are better controlled.
3. *Level 3 — Qualitative*
   Costs and schedules are more tightly controlled but quality is still variable.
4. *Level 4 — Quantitative*
   Reasonable control over cost, schedule, and quality.
5. *Level 5 — Integrated*
   Quantitative basis for continual improvement of cost, quality, and responsibility.

This attempt does lack any meaningful software measurement related best practices and even the scientific rigor. So, it will be entirely omitted.

4.3.4 Mendonça's approach to improving existing measurement frameworks

In his PhD thesis, Mendonça [Men97] seizes the question of how to improve existing measurement frameworks and proposes a measurement process model that integrates both paradigms, the top-down and the bottom-up, in his own way . The iterative model consists of the three different phases: First, the measurement framework shall be characterized in order to ease the understanding of the data user's measurement information needs. In doing so, structured interviews and document reviews aid in identifying key components such as software measures, attributes, existing data, user groups, data uses. Second, a top-down analysis shall be conducted on the lines of the GQM to enable a judgment, how well the legacy data that is already being collected in the organization satisfies the elicited needs. Third and finally, a bottom-up analysis of the legacy, genuine data shall be conducted to extract novel information. Although Mendonça limits his model to the attribute focusing data mining technique, it seems to be applicable using any other analysis techniques, as well. However, the bottom-up processes of phase three would then need to be changed, accordingly. The activities for each of the three phases are listed below [Men97, pp. 35]:

1. **"Measurement framework characterization**

   (a) Identify metrics
   (Review available measurement or software process documents.)

(b) Identify available data
(Use descriptions of software measures and review data repositories' documentation. Examine data repositories directly and interview data managers responsible for the repositories if necessary.)

(c) Identify data uses and user groups
(Interview data managers. Talk to specific user group representatives.)

(d) Identify attributes
(Interview data user representatives or, if possible, extract the description directly from the measurement framework documentation.)

2. **Top-down analysis (GQM)**

(a) Capture data user goals
(Interview group representatives. Use a GQM goal template.)

(b) Identify relevant entities
(Interview the group representatives and/or, if possible, extract the description directly from available documentation about the object of study.)

(c) Identify relevant attributes
(Extract the comprehensive list of attributes directly from available documentation or use the existing attribute lists. Interview the group representatives to produce an updated list of relevant attributes. Ask them to rate the importance of each attribute. Describe the attribute implicitly, textually, or formally.)

(d) Map attributes to existing metrics
(Establish the mapping between metrics and relevant attributes by comparing the description of relevant attributes with the attributes associated with the existing metrics. Interview data user group representatives to validate if the mapped metrics are consistent with the user goals.)

3. **Bottom-up analysis**

(a) Establish relationship questions
(Interview the group representatives or use the group goals and description of available data to determine the generic relationship questions.)

(b) Define the analysis
(Interview the group representatives or use the group goals to determine the analysis scope and granularity. Use description of available data, metrics, and the analysis scope to extract the data. Use the description of the analysis granularity to pre-process the data.)

(c) Run the analysis
(Use the generic relationship questions to identify metric groups (each attribute class corresponds to a group). Manipulate the cutoff and maximum number of diagrams to obtain a reasonable number of 'interesting' diagrams. Set the analysis dimension based on the number of attribute classes.)

(d) Organize the diagrams
(Review available diagrams one by one. Discard useless diagrams and organize the others using some consistent criteria.)

(e) Review the diagrams
(Review diagrams together with data users or managers.)"

Contents perspective

Criterion C1: With the focus of this approach being on improving existing SMPs — or as the author calls 'measurement frameworks' — the criterion is fully satisfied.

Criterion C2: The model is based on a number of assumptions that have been derived from literature and practice. Because the model could expose its usefulness and the absence of flaws during evaluation in practice [Men97] [MB00], also this criterion is satisfied.

Criterion C3: In his own manner, Mendonça integrates the top-down and bottom-up paradigm in his approach into a mixed model. However, the entry level of software measurement represented by the a posteriori or a priori bottom-up paradigm is not reflected. Mendonça and Basili [MB00] bring forward the following: "It is aimed at applying the principles of goal-oriented measurement in an environment that is already functional, instead the more common view of defining a measurement process from scratch based on the measurement goals of data users." Because of that omission, this criterion is not satisfied.

Model perspective

Criterion M1: The process model underlying that improvement approach for SMPs does provide entry criteria, inputs, procedures, methodologies, outputs, and exit criteria. These are similar to the description of purposes and outcomes as demanded by ISO/IEC Standard 15504. So this criterion is satisfied.

Criterion M2 – M4: Owing to the fact that the author of this model did not intend to assess the capabilities to implement and sustain SMPs, the aspect of assessing the capabilities has been disregarded during its development. So, this criteria are apparently not fulfilled.

Criterion M5: Since the improvement approach of Mendonça is based on a three-part process model that highlights a straightforward path for improvement, this criterion is satisfied.

4.3.5 The Measurement-CMM

As part of the PhD research of Niessink the Measurement-CMM (M-CMM) [NvV98] [Nie00] was developed to assess and — based on that assessment — to improve the capabilities of organizations in terms of software measurement. In contrast to its dedicated archetype, that is the CMM of SEI, the focus of this model does not lay in the capability of an organization's software development process but in its 'measurement capability', Niessink [Nie00, p. 83] defines as: "the extent to which an organization is able to take relevant measures of its products, processes and resources in a cost effective way resulting in information needed to reach its business goals."

Having been rudimentarily designed on the lines of the CMM, the M-CMM avails itself of two basic structural elements, that is, KPAs are assigned to capability maturity levels denominated as in the CMM according to their increase of sophistication. As indicated by Niessink himself, the model lacks a description of practices for each KPA as well as a questionnaire for them. It turns out to be in large parts incomplete and scientifically shallow. However, the assignment of key process areas to maturity levels as word-by-word taken from Niessink [Nie00, pp. 85] is presented next:

1. **"Initial:**

   - No KPAs

2. **Repeatable:**

   - *Measurement design:* Measurement goals, measures and measurement protocols are established according to a documented procedure, and goals, measures and protocols are kept consistent with each other. Measurement protocols are managed and controlled.
   - *Measure collection:* Measures are collected according to the measurement protocol.
   - *Measure analysis:* The collected measures are analyzed with respect to the measurement goals.
   - *Measurement feedback:* The measurement goals, the measurement protocols, the collected measures and the results of the analysis are made available to the people involved in the measurement process.

3. **Defined:**

   - *Organization measurement focus:* Software measurement activities are coordinated across the organization. Strengths and weaknesses of the measurement process are identified and related to the standard measurement process.
   - *Organization measurement design:* A standard measurement process for the organization is developed and maintained and information with respect to the use of the standard measurement process is collected, reviewed and made available.
   - *Organization measure database:* Collected measures are stored in an organization-wide database and made available.
   - *Training program:* People are provided with the skills and knowledge needed to perform their roles.

4. **Managed:**

   - *Measurement cost management:* Costs of measurement are known and used to guide the 'measurement design' process and the 'organization measurement design' process.
   - *Technology selection:* The information of measurement costs is used to choose and evaluate technology support for the measurement process.

5. **Optimizing:**

   - *Measurement change management:* The measurement capability is constantly being improved by monitoring the measurement processes and by anticipating changes in the software process or its environment."

Contents perspective

Criterion C1: The model has been designed to assess and improvement an organization's capabilities with special respect to measurement. This criterion is fully satisfied.

Criterion C2: Although the model has been developed in the course of a research project, the expected degree of scientific rigor could not been found during the evaluation. Neither the related papers nor the PhD thesis provides information on the foundation, on which the model's elements were extracted and aligned. It is for instance questionable to locate the elementary training process for software measurement on level three instead on level one. Since the justifications for a number of those issues is not given, the lack of scientific rigor leads to the failure of this criterion.

Criterion C3: The KPAs assigned to the pertinent levels of software measurement capability maturity do neither reflect best practices as elicited in subsection 2.8.4 nor do they dwell on different paradigms of software measurement. Also, this criterion fails.

Model perspective

Criterion M1 – M5: Because the model's development has been scientifically shallow and rudimentarily, it did not take account for any model-related aspects. This fact is reflect in the negative evaluation of all model-related criteria.

4.4 Conclusion

Within this chapter the brief review and evaluation of SPA/SPI models implicitly and several models explicitly dealing with the implementation and sustainment of the software measurement process was presented to answer the third sub-question of this research work:

> *RQ3. Find and evaluate current process improvement models that deal with the implementation of the software measurement process using criteria from RQ1/RQ2! Expose a basis model, that satisfies at least a large share of the criteria, together with its specific shortcomings!*

Results of the evaluation

In the course of the investigations it became obvious that only two of the three proposed implicit and two of the four explicit candidate models exhibit meaningfulness or sufficient material for an observation according to the set context. The results of the evaluations with respect to the criteria developed in the previous chapters shall be summarized with the aid of the following table 4.3. It can be easily recognized that none of the models fully satisfies all of the set criteria.

Model	Decision criteria							
	C1	C2	C3	M1	M2	M3	M4	M5
Implicit coverage:								
- CMM v1.1	–	–	–	–	–	–	✓	✓
- ISO/IEC Standard 9001:2000	*Disregarded from evaluation*							
- CMMI Framework v1.1	✓	✓	–	✓	✓	✓	✓	✓
Explicit coverage:								
- Software measurement technology maturity	✓	–	–	–	–	–	–	–
- Measurement maturity model	*Disregarded from evaluation*							
- META Measurement Maturity model (4M)	*Disregarded from evaluation*							
- Mendonça's approach to improving existing measurement frameworks	✓	✓	–	✓	–	–	–	✓
- Measurement-CMM (M-CMM)	✓	–	–	–	–	–	–	–

Legend: ✓ = Satisfied | – = Not satisfied

Table 4.3: Results of the observation of existing methods with implicit or explicit coverage

Exposition of the basis model and its shortcomings
The implicit software measurement process improvement model covered in the CMMI-MA support process area of CMMI Framework v1.1 is second to none, because it solitarily satisfies a large share of the evaluation criteria. However, it shipwrecks and shows a shortcoming when it comes to the support of the sequence of measurement paradigms starting from bottom-up over mixed to top-down during software measurement process implementation.

Basis model:

B1 'Measurement & Analysis' (CMMI-MA) support process area of the CMMI Framework v1.1

Shortcoming to be addressed:

*B2 Lack of support of the sequential application of the 'bottom-up', 'mixed', and 'top-down' **measurement paradigms** during implementation of the software measurement process.*

Part II

Proposal and development of a better solution

Chapter 5

The Software Measurement Process Improvement (SMPI) model

"If we had management that knew what the right goals and questions to ask were, we wouldn't need better measurement as badly as we do!"

– Bill Hetzel [Het93, p. 25] –*

5.1 Introduction

In the previous chapter it has been shown by the author that neither mainstream SPA/SPI models, which implicitly cover aspects of SMP implementation and sustainment processes, nor those models that have been explicitly designed for this purpose, satisfy the criteria imposed. More specifically, the evaluation of the previous chapter revealed that the CMMI-MA process area of the CMMI Framework v1.1 exemplarily implements the related, up-to-date standards. That is, it fully complies with the software measurement process model of ISO/IEC Standard 15939 from the contents perspective and, roughly speaking, with the regulations of ISO/IEC Standard 15504 (SPICE) from the model perspective. Because of that, the **CMMI-MA is preferred as basis model**. But as suggested in chapter 2, the ISO/IEC Standard 15939 does exclusively address the top-down paradigm for software measurement and omits an exhaustive coverage of the bottom-up and mixed paradigms; and so the CMMI does, as well. Thus, the evaluation was positive for all but criterion **C3** that **turned out to be a shortcoming**. For reasons of transparency the evaluation criteria and results of the previous chapters are compiled, once more:

	RQ1 (Content-related criteria)	*CMMI-MA*
C1.	The **scope** of the process improvement model must be (at least partially) on software measurement process implementation in industrial settings.	✓
C2.	The process improvement model must have been developed with **scientific rigor**.	✓

*Software measurement and testing guru; fellow of Software Practices Research Center at Jacksonville, FL, USA

C3.	The process improvement model must be able to reflect the sequential application of the 'bottom-up', 'mixed', and 'top-down' **measurement paradigms** during implementation of software measurement processes in industrial settings.	–
	RQ2 (Model-related criteria)	*CMMI-MA*
M1.	The process improvement model must provide a **process reference model** compliant with the requirements of ISO/IEC Standard 15504.	✓
M2.	The process improvement model must be **compatible with the measurement framework** of ISO/IEC Standard 15504.	✓
M3.	The process improvement model's **process assessment model** must represent a mapping of the process reference model and the measurement framework of ISO/IEC Standard 15504.	✓
M4.	The process improvement model's **assessment process** must observe the regulations of ISO/IEC Standard 15504.	✓
M5.	The process improvement model must provide process **improvement guidelines** for the specified scope.	✓

5.2 Concept and design of the complemented model

According to Comer and Chard [CC93] the maturity of an organization's software measurement process is directly correlated to the ability to define adequate business and/or software measurement goals. This implies an improvement in terms of the used software measurement paradigms from less goal-oriented (bottom-up paradigm) over more goal-oriented (mixed paradigm), to entirely goal-oriented (top-down). Such a shift is neither considered in prevalent software measurement process models nor in related process improvement models. But as Fuchs [Fuc95, p. 75] admonishes: "Only in a reasonably mature development environment can a top-down approach be expected to work. The majority of companies will be at a low level of maturity and should avoid starting with a top-down approach ... companies having already started a measurement programme should try to move towards a mixed approach as a first step ... and not directly towards a top-down one." To put things right, the solution of CMMI-MA process area selected as basis model, shall be complemented to that effect towards the SMPI model.

5.2.1 The development concept

Based on the results of the evaluation recapitulated in the introduction of this chapter, the shortcoming of the selected basis model, CMMI-MA, in terms of criterion C3 can be paraphrased to the following development concept (DC):

> *DC. Extend the 'Measurement & Analysis' support process area of the CMMI Framework v1.1 so that it can support the sequential application of the 'bottom-up', 'mixed', and 'top-down'* ***measurement paradigms*** *during implementation of the software measurement process in industrial settings!*

5.2.2 The design and development rationale

The rationale behind the improvement approach is to keep the general contents and structure of the CMMI Framework v1.1 because it satisfactorily meets the model-related criteria extracted in chapter 3, but to improve the revealed lack of paradigm shift of the software measurement process as manifested in the CMMI-MA support process area.

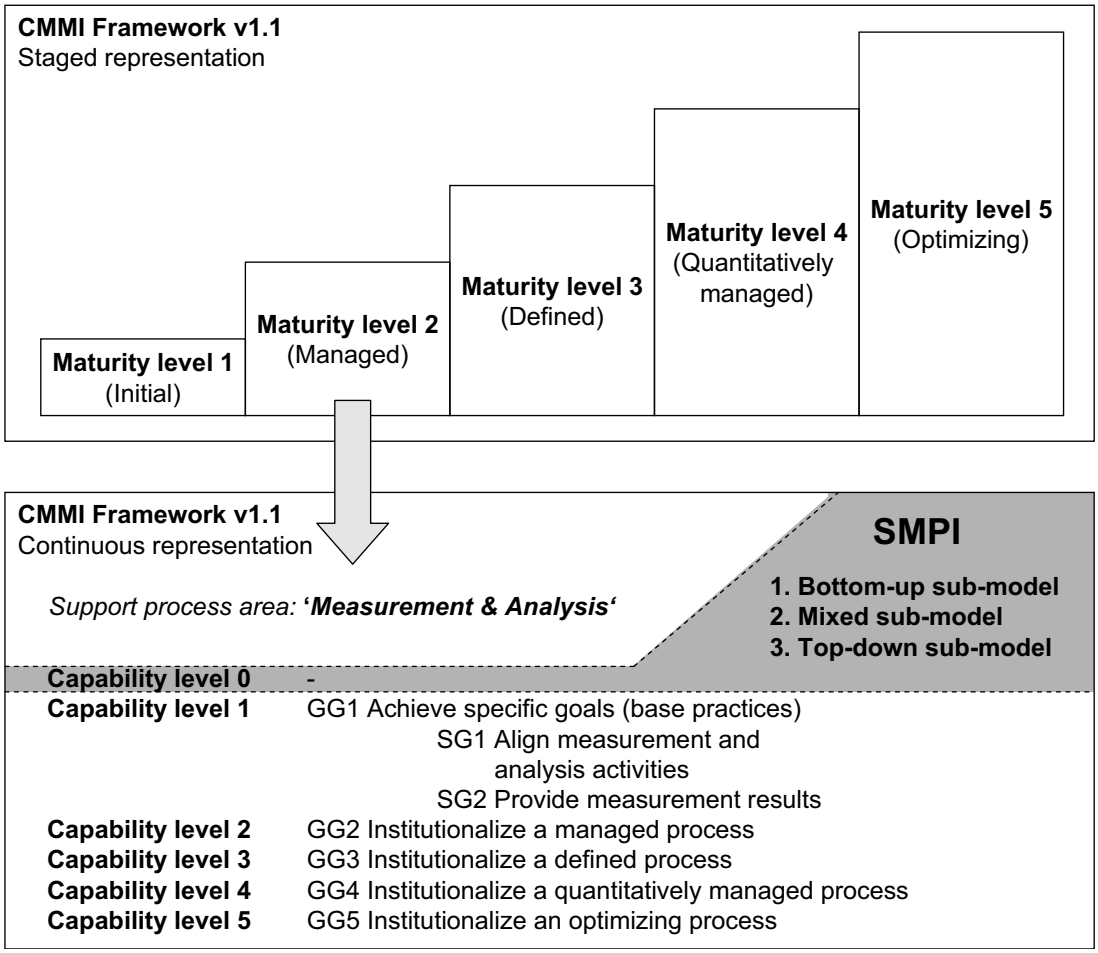

Figure 5.1: The design rationale behind the SMPI model

As illustrated in figure 5.1, this includes leaving the assignment of the otherwise well-designed process area unchanged at maturity level two in the staged representation and allow the improvement of that process area along the process capability levels zero to five in the continuous representation, as established.

Given the continuous representation, the contempt of the software measurement paradigm shift outcrops as early as in capability level one of the process area: It has the generic goal to 'Achieve specific goals' by conducting the base practices of the top-down paradigm connected with specific goals one ('Align measurement and analysis activities') and two ('Provide measurement results'). This would not be a mistake if there only was one paradigm to observe. As shown before, goal-oriented measurement is indeed a sublime goal, but already mirrors a state of fundamental experiences gained by running through the software measurement paradigm shift.

Keeping in mind the intention to leave the general contents and structure of the CMMI unchanged while adding essential processes for software measurement according to the bottom-up and mixed paradigms, which have been obviously omitted, as well as the well-established top-down paradigm, a supplemental process model covering the progress through these measurement paradigms should be developed. Such a model can ensure that a practicable course of improvement of the software measurement process can be reproduced without having the need to develop the model from scratch with high costs. [FLZ05] Moreover, by incorporating best practices into the reference model, 'best-of-breed' knowledge is manifested and shared among the community. Insofar, reference models provide a sound foundation for the creation of models tailored to specific conditions [Tho05] such as an organizational context. Owing to the fact, that this model is thought to facilitate stepwise software measurement process improvement along the measurement paradigms as presented in section 2.5, it should bear the name *Software Measurement Process Improvement (SMPI)*.

With respect to the development concept (DC), the design and development rationale (DR1, DR2, DR3) for the complemented SMPI model can be summarized from the above remarks as the following procedure:

DR1. Establish consensus among the phases of the different software measurement process models specific for each measurement paradigm.

DR2. Perform a mapping of processes and activities onto the consensus phases of the different software measurement process models specific for each paradigm. Bring life into topics not covered.

DR3. Provide a graphical and a textual formulation for each of the different software measurement process models specific for each measurement paradigm.

These parts of the development rationale will be worked off one by one in the following.

5.3 Development of the complemented model

5.3.1 Consensus of measurement paradigm-specific process phases

With respect to the formal definition of the core software measurement process (MP) in equation 2.7, several procedures and/or practices ($P_{materialization}, P_{collection}, P_{analysis}, P_{exploitation}$) have to be performed to conduct this process. As could be easily seen from previous remarks (especially in section 2.5), different measurement paradigms exist that strongly influence the course of action to be taken. In order to reveal a consensus between the building blocks of the major process suggestions, which is the case for software measurement according to the bottom-up and mixed measurement paradigms, or measurement process models for the top-down paradigm, the respectively proposed process phases have been confronted in tables 5.1, 5.2, and 5.3 with the respective consensus phases printed in the first column.

	Phases of the measurement process in reference material		
Bottom-up model phases of SMPI	Hetzel et al. [Het93]	Bache et al. [BN95]	ISO/IEC Standard 15939 [ISO02]
Selection of software measures	1. Definition of software measures	0. Process investigation 1. Selection of software measures	2. Plan the measurement process (general part)
Data collection	2. Data collection	2. Data collection	3. Perform the measurement process
Data analysis	3. Analysis and modeling	3. Analysis	
Causal analysis	4. Question triggering goal deduction	4. Causal study, goal setting	

Table 5.1: Consensus of bottom-up software measurement process models phases

	Phases of the measurement process in reference material		
Top-down model phases of SMPI	Gresse et al. [GHW95]	Solingen et al. [vSB99]	ISO/IEC Standard 15939 [ISO02]
Goal materialization to software measures	1. Prestudy 2. Identify GQM goals 3. Produce GQM plan	2. Definition	2. Plan the measurement process
Data collection	4. Collect and validate data	3. Data collection	3. Perform the measurement process
Data analysis	5. Analyze data	4. Interpretation	

Table 5.2: Consensus of top-down software measurement process models phases

Mixed model phases of SMPI	**Phases of the measurement process in reference material**			
	Fuchs [Fuc95]	Mendonça [Men97]	Ramil et al. [RL01]	ISO/IEC Standard 15939 [ISO02]
Goal materialization to software measures	1. Top-down definition of software measures	1. Measurement framework characterization 2. Top-down analysis	1. Top-down definition of software measures	2. Plan the measurement process
Selection of software measures	2. Selection of software measures		2. Bottom-up selection of software measures	2. Plan the measurement process (general part)
Reconciliation	3. Reconciliation between sets of measures			
Data collection				3. Perform the measurement process
Data analysis		3. Bottom-up analysis		

Table 5.3: Consensus of mixed software measurement process models phases

5.3.2 Imbueing the process models' phases with life

Subsequent to the derivation of process phases for the three, paradigm-specific measurement process models by consensus the contents (activities) of each phase for each sub-model needs to be defined. Once more, this activity is characterized by a consensus-based mapping procedure from the reference material, especially of the 'Measurement & Analysis' process area of the CMMI Framework v1.1 as well as the applicable part of best practices for SMP implementation and sustainment as revealed in section 2.8.4.

Since the Root Cause Analysis (RCA) methodology, which is predestinated to provide activities for the phase 'Causal analysis' of the bottom-up sub-model of the SMPI, has been treated stepmotherly by both, the reference material as well as the best practices, additional and sound knowledge of its procedural steps as per Wilson et al. [WDA93], Rooney and Vanden Heuvel [RH04], or Jucan [Juc05] has been quested and used. The possibly arising and undoubtedly legitimate critic that RCA (more specifically CMMI's process area Causal Analysis & Resolution (CAR) assigned to maturity level five of the staged representation) could be off scope for process areas required earlier, such as 'Measurement & Analysis', can be answered back: The author of this thesis feels vindicated because Buglione and Abran [BA06] recently proposed the relocation of the CAR process area to CMMI's maturity level two as a basic support process.

5.4 Presentation of the SMPI model

Having the aim to provide an empirical, descriptive software measurement process improvement model covering the improvement steps along the distinct measurement paradigms from the functional, behavioral, organizational, and informational perspectives for the management domain (cf. section 3.2), both, a graphical and a textual formulation using two different PDLs is preferred to accentuate the purpose of describing a process to easily communicate it to other people. [Oul95]

Based on a comparison between different graphical PDLs, e. g. Nysetvold and Krogstie [NK05] found out that especially the Business Process Modeling Notation (BPMN) of the Object Management Group (OMG) [Gro06] comes off well. Moreover, recently List et al. [LK06] provided evidence of the multiple advantages of that notation for the modeling of business processes. Because the core software measurement process can be also regarded as a business process, the BPMN shall be used to model diagrams.

According to a survey of Paulk et al. [PGW01] most high maturity organizations use either the Entry criteria – Tasks – Validation – Exit criteria (ETVX) process modeling methodology of Radice and Phillips [RP88] or the Entry criteria – Inputs – Tasks – Validation – Outputs – Exit criteria (EITVOX) being an offshoot of ETVX methodology that has been proposed by the SEI [aC94]. While ETVX addresses the basic set of mandatory information needed to perform a work activity independent from the level of abstraction with nesting being possible [PGW94], EITVOX has been enhanced by helpful information concerning the inputs and the outputs of the work activity as illustrated in figure 5.2. Having both shortlisted, the choice fell on the EITVOX methodology to describe the sub-models and their constituents in detail.

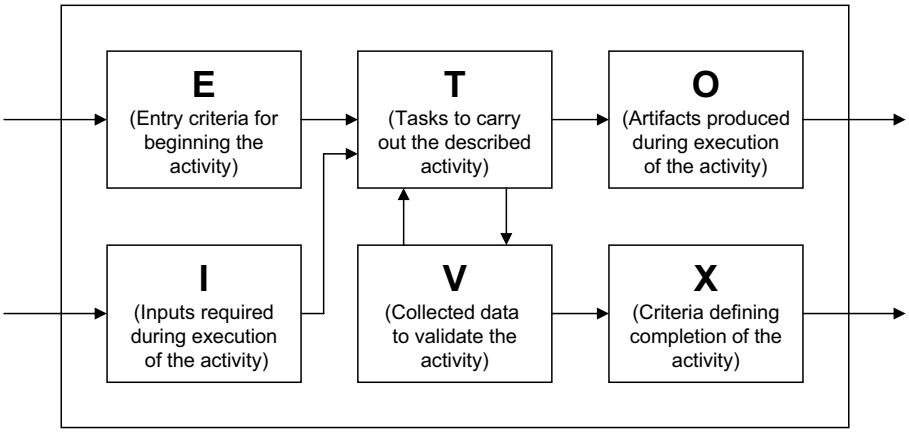

Figure 5.2: The elements of SEI's EITVOX process modeling paradigm (adapted from [aC94])

5.4.1 The whole model at a glance

From a bird's eye view, the SMPI provides the three paradigm-specific sub-models bottom-up ($P_{Bottom-up} = P_B$), mixed ($P_{Mixed} = P_M$), and top-down ($P_{Top-down} = P_T$) that provide different activities within the software measurement process improvement model depending on the starting conditions.

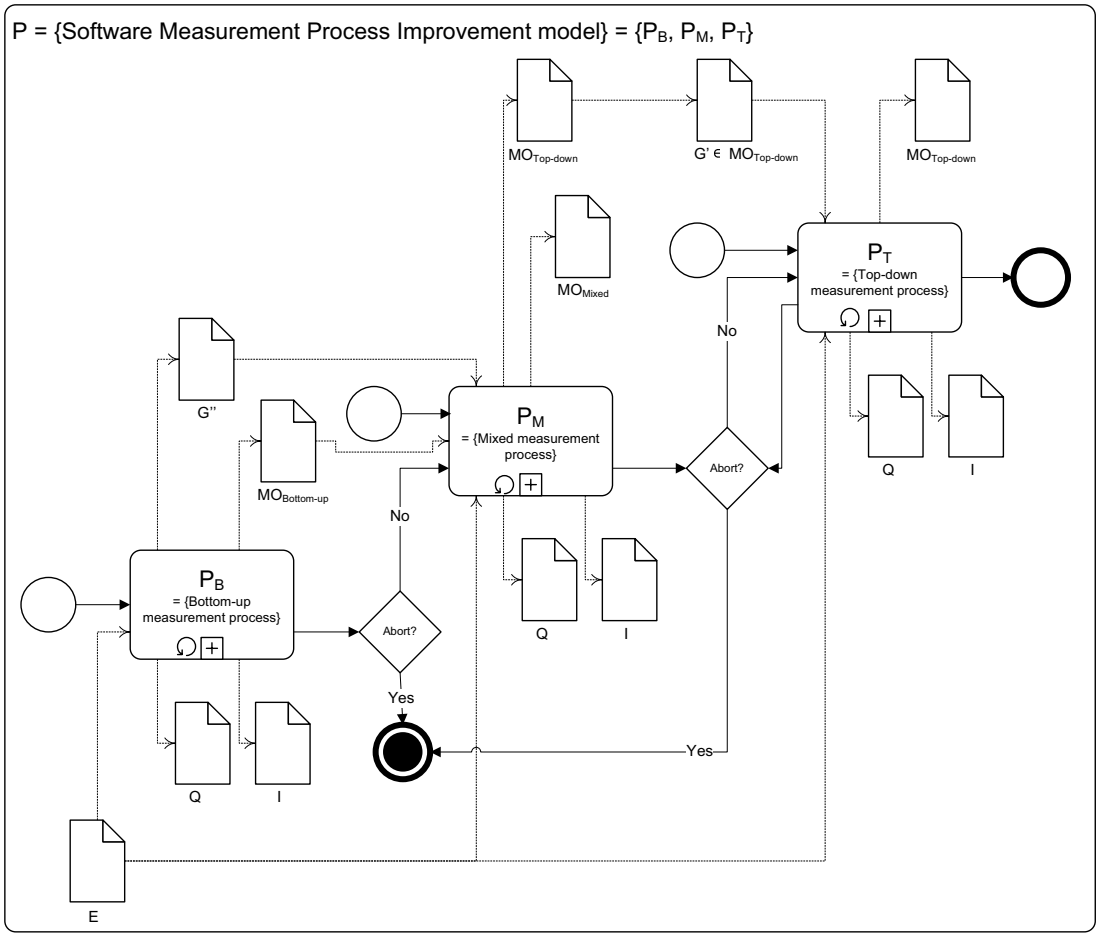

Figure 5.3: The SMPI model at a glance

The general structure of the SMPI model is illustrated in figure 5.3 using the graphical elements of BPMN. Here, the bottom line is that the general aim of each sub-model is to quantify software engineering entities (E) into quantities (Q), which are then transformed into measurement information products (I). Depending on an organization's ability to solely adopt foreign software measures or to define measurement goals, the processes of the sub-models produce measurement outlines (MO). These measurement outlines comprise the set of established measurement thresholds and/or measurement goals (G'), the selected software engineering entities (E'), their attributes (A) quantified by software measures (M), operational definitions (O) as well as data collection and analysis directives (D) and the related responsible personnel (R).

Another aspect of the model as a whole is the fact that an organization can directly jump to the sub-model being most suitable in dependence of the requirements (entry criteria) of each sub-model. Moreover, it is also possible to iteratively conduct the processes of a sub-model or to abort it after each iteration.

5.4.2 P_B — The bottom-up sub-model in detail

If a SMP has never been in place before in an organization and it is about to start — driven by its own, unsettled measurement interest or it has been urged from external side to jump on the bandwagon — the most suitable starting point would definitively be the *a priori* bottom-up sub-model (P_B) as depicted in figure 5.4. However, this presumes that the measurement initiative was set up as a properly staffed, well-funded project having experienced personnel. An improvement would then take place by proceeding through the mixed and later the top-down sub-models, one after another.

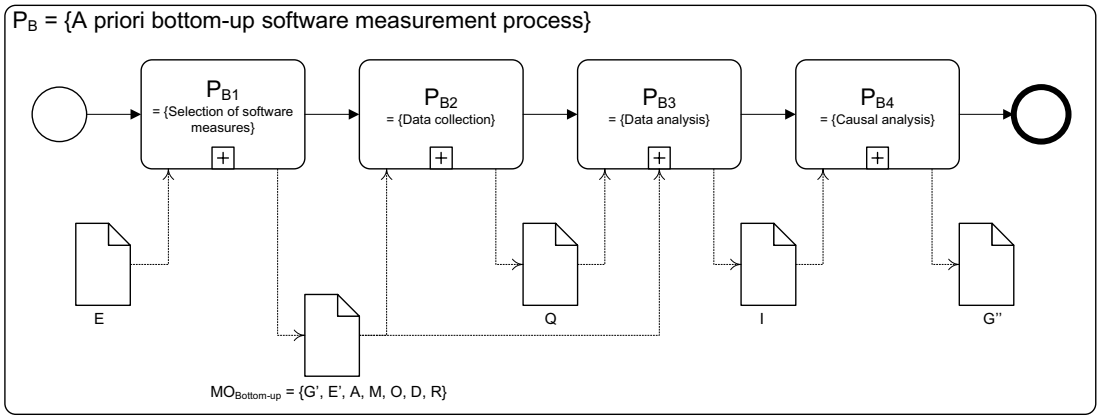

Figure 5.4: BPMN diagram of P_B

The first step in the bottom-up sub-model is the selection of software measures (P_{B1}) for software engineering entities of interest (E) despite having no predefined measurement goals. This results in a bottom-up measurement outline ($MO_{Bottom-up}$), which is then used as foundation for the second step of data collection (P_{B2}). Together with the measurement outline the collected quantities (Q) serve as input for the subsequent data analysis (P_{B3}) intended to produce the measurement information products(I). Due to the absence of real measurement goals, which have been substituted in step one by thresholds (G') for the adopted software measures, the step of causal analysis (P_{B4}) to derive real measurement goals (G'') required for the improvement of the software measurement process has to go along. Using the EITVOX process modeling paradigm, the situation is as following:

$P_{Bottom-up} = P_B$		
Entry criteria:	-	$G = \emptyset$
	-	$MO_{Bottom-up} = \emptyset$
	-	$E \neq \emptyset$
Inputs:	-	E
Tasks:	-	$\mathbf{P_{B1}} = \{\text{Selection of software measures}\}$
	-	$\mathbf{P_{B2}} = \{\text{Data collection}\}$
	-	$\mathbf{P_{B3}} = \{\text{Data analysis}\}$
	-	$\mathbf{P_{B4}} = \{\text{Causal analysis}\}$
Validation data:	-	The element counts of the subsets of $MO_{Bottom-up}$
	-	The element counts of the sets Q, I, G''
Outputs:	-	$MO_{Bottom-up}, Q, I, G''$
E**X**it criteria:	-	$G'' \neq \emptyset, I \neq \emptyset$

5.4.2.1 Examining the process group P_{B1} at close quarters

To select meaningful software measures based on the software engineering entities (E) at hand, a universe of software measures should be built ($P_{B1,1}$). Together with these measures (M), an initial share of the bottom-up measurement outline ($MO_{Bottom-up}$) consisting of the software engineering entities (E') and their attributes (A) to quantify as well as operational definitions (O), can be prepared. Afterwards and depending on the selected software measures, thresholds are revealed and imposed ($P_{B1,2}$) as measurement goals (G') failing which top-down defined ones. Having done this, data collection and storage procedures ($D_{Collection}$) and the responsible personnel ($R_{Collection}$) can be specified ($P_{B1,3}$). Furthermore, in a final process step, the analysis procedures ($D_{Analysis}$) and the personnel responsible for the conduct of the analysis ($R_{Analysis}$) should be specified ($P_{B1,4}$). Ultimately, the result should be a complete and available bottom-up measurement outline as formulated according to the EITVOX process modeling paradigm beneath and illustrated in figure 5.5:

		$P_{B1} = \{\textbf{Selection of software measures}\}$
Entry criteria:	-	$G = \emptyset$
	-	$MO_{Bottom-up} = \emptyset$
	-	$E \neq \emptyset$
Inputs:	-	E
Tasks:	-	$\mathbf{P_{B1,1}} = \{\text{Build a universe of software measures}\}$
	-	$\mathbf{P_{B1,2}} = \{\text{Impose thresholds as measurement goals}\}$
	-	$\mathbf{P_{B1,3}} = \{\text{Specify data collection/storage procedures}\}$
	-	$\mathbf{P_{B1,4}} = \{\text{Specify analysis procedures}\}$
Validation data:	-	The element counts of the subsets of the measurement outline. ($\{G', E', A, M, O, D, R\} \subset MO_{Bottom-up}$)
Outputs:	-	$MO_{Bottom-up}$
E**X**it criteria:	-	$G' \neq \emptyset; E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset; D \neq \emptyset; R \neq \emptyset$

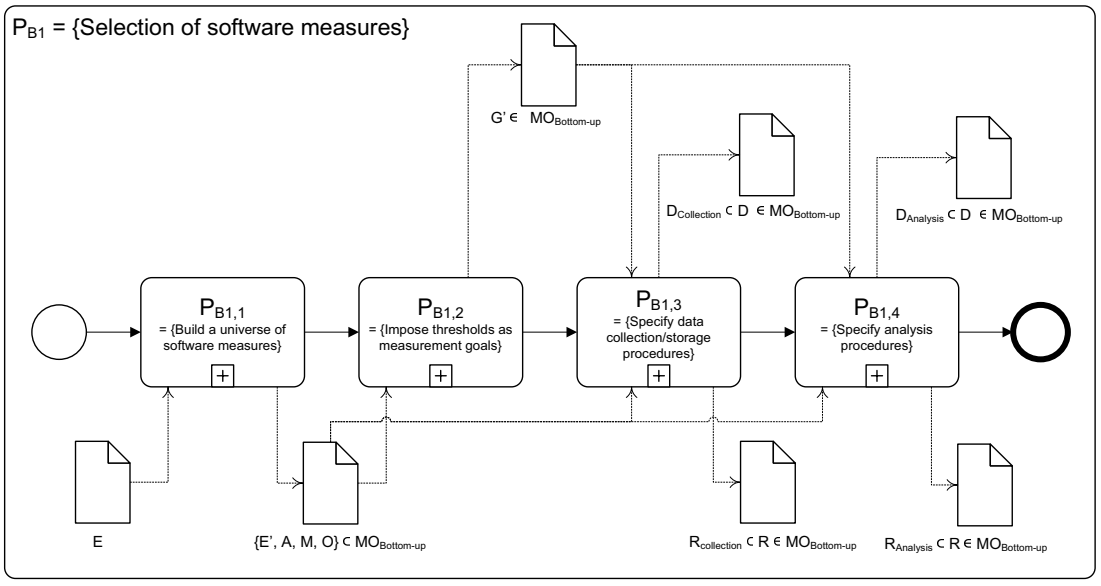

Figure 5.5: BPMN diagram of P_{B1}

The inner life of process $P_{B1,1}$
The first activity ($P_{B1,1-1}$) of this process requires one to identify the software engineering entities of interest (E') from the general ones (E). Only for these entities of interest a list of candidate software measures is produced ($P_{B1,1-2}$), this list is balanced ($P_{B1,1-1}$) and base measures (M) from the candidate ones are selected ($P_{B1,1-4}$). Once, the measures have been chosen, the attributes (A) that are observed by the selected software measures can be derived ($P_{B1,1-5}$) and operational definitions (O) specified ($P_{B1,1-6}$). After all, the results of the process should be documented and publicized ($P_{B1,1-7}$).

The following remarks according to the EITVOX process modeling paradigm and the BPMN diagram in figure 5.6 describe the activities, as well:

$P_{B1,1} = \{\textbf{Build a universe of software measures}\}$		
Entry criteria:	-	$G = \emptyset$
	-	$MO_{Bottom-up} = \emptyset$
	-	$E \neq \emptyset$
Inputs:	-	E
Tasks:	-	$\mathbf{P_{B1,1-1}} = \{\text{Identify software engineering entities of interest.}\}$
	-	$\mathbf{P_{B1,1-2}} = \{\text{Prepare a list of well} - \text{established candidate software measures.}\}$
	-	$\mathbf{P_{B1,1-3}} = \{\text{Balance the list of candidate measures with available software engineering entities.}\}$
	-	$\mathbf{P_{B1,1-4}} = \{\text{Select base measures from the candidate ones.}\}$
	-	$\mathbf{P_{B1,1-5}} = \{\text{Derive the attributes that are observed by the chosen software measures.}\}$
	-	$\mathbf{P_{B1,1-6}} = \{\text{Specify operational definitions for the selected software measures.}\}$
	-	$\mathbf{P_{B1,1-7}} = \{\text{Document results of the process and publicize to all involved stakeholders.}\}$
Validation data:	-	The element counts of the subsets of the measurement outline. ($\{E', A, M, O\} \subset MO_{Bottom-up}$)
Outputs:	-	E'
	-	List of candidate software measures
	-	Balanced list of candidate software measures
	-	A
	-	M
	-	O
E**X**it criteria:	-	$E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset$

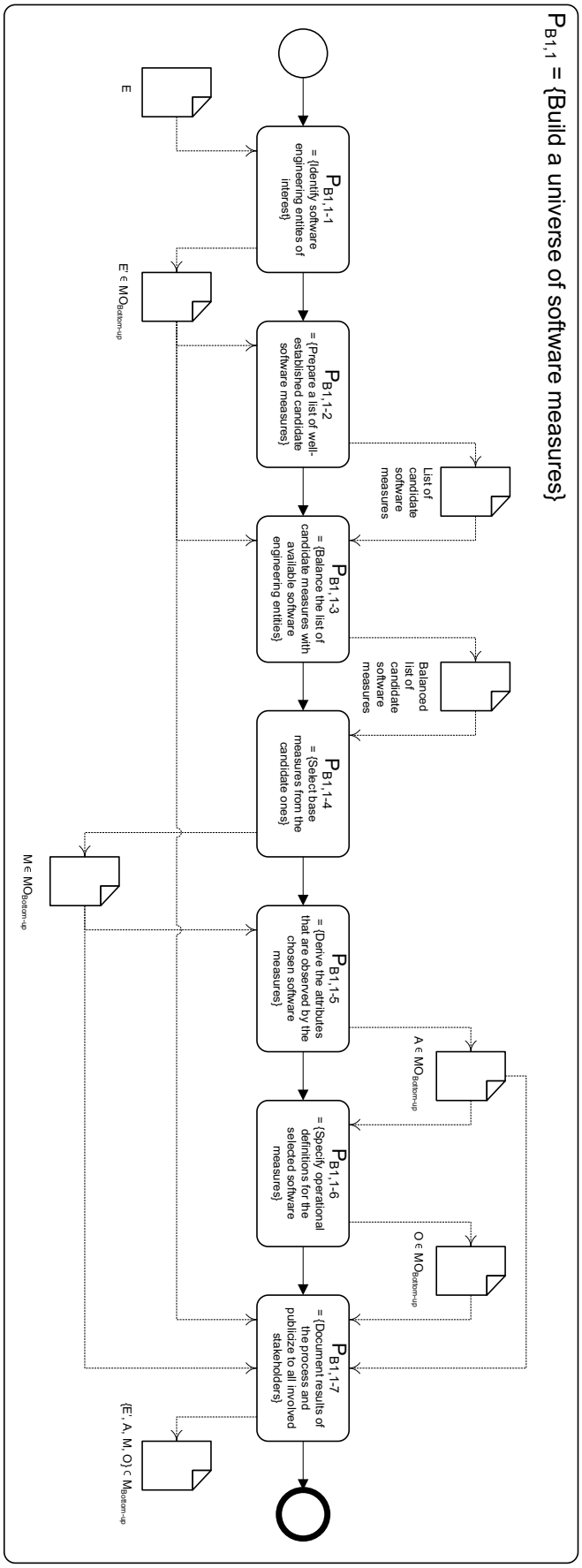

Figure 5.6: BPMN diagram of $P_{B1,1}$

The inner life of process $P_{B1,2}$

Taking the completed bottom-up definition of software engineering entities of interest (E'), attributes (A), software measures (M), and operational definitions (O) for granted, a list of candidate thresholds for the selected software measures can be identified ($P_{B1,2-1}$). Form that list, specific thresholds should be reduced ($P_{B1,2-2}$) to a balanced list of candidate thresholds. Next, a sound argumentation that justifies imposing the selected thresholds as measurement goals failing which real, top-down defined ones, should be given ($P_{B1,2-3}$). Afterwards, the selected thresholds should be defined ($P_{B1,2-4}$) as measurement goals (G') and the results of the process, that is, the argumentation and the set of proxy measurement goals, should be documented and publicized. ($P_{B1,2-5}$) That context is also described in the table beneath and in figure 5.7:

$P_{B1,2} = \{$**Impose thresholds as measurement goals**$\}$		
Entry criteria:	-	$E' \neq \emptyset, A \neq \emptyset, M \neq \emptyset, O \neq \emptyset$
Inputs:	-	$\{E, A, M, T, F\} \subset MO_{Bottom-up}$
Tasks:	-	$\mathbf{P_{B1,2-1}} = \{$Identify candidate thresholds for the selected software measures.$\}$
	-	$\mathbf{P_{B1,2-2}} = \{$Select specific thresholds from the list of candidate thresholds.$\}$
	-	$\mathbf{P_{B1,2-3}} = \{$Provide a sound argumentation that justifies imposing the selected thresholds as measurement goals.$\}$
	-	$\mathbf{P_{B1,2-4}} = \{$Define the selected tresholds as measurement goals.$\}$
	-	$\mathbf{P_{B1,2-5}} = \{$Document results of the process and publicize to all involved stakeholders.$\}$
Validation data:	-	The element counts of the subset of established goals of the measurement outline. ($\{G'\} \subset MO$)
Outputs:	-	List of candidate thresholds
	-	Balanced list of candidate thresholds
	-	Argumentation
	-	G'
E**X**it criteria:	-	$G' \neq \emptyset$

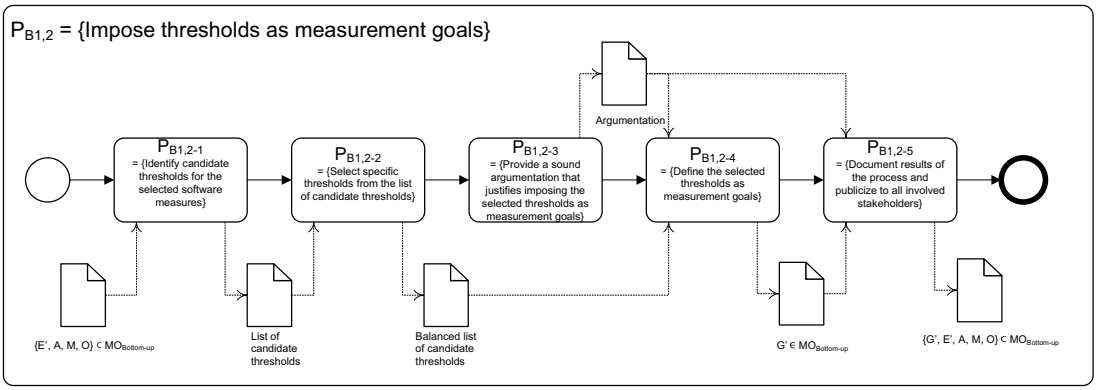

Figure 5.7: BPMN diagram of $P_{B1,2}$

The inner life of process $P_{B1,3}$

After identifying ($P_{B1,3-1}$) a list of existing sources of data, that list should be used together with the results of the previous process step ($\{G', E', A, M, O\}$) to identify a list of measures for which data are needed, but are not currently available ($P_{B1,3-2}$) and to specify, how to collect and store data ($D_{Collection}, R_{Collection}$) for each required software measure ($P_{B1,3-3}$). Based on these information, data collection mechanisms and process guidance ($D_{Collection}$) for the responsible personnel can be created ($P_{B1,3-4}$). Then, the automatic collection of data where appropriate and feasible should be focused on ($P_{B1,3-5}$). After all, the procedures for data collection and storage as well as the data on responsible personnel should be prioritized, reviewed, and updated ($P_{B1,3-6}$). Should arise a need to align software measures and measurement goals due to the data collection and storage procedures, this ($P_{B1,3-7}$) should be done on the results ($\{G', E', A, M, O\}$) of the previous process step. That context is also described in the table beneath and in figure 5.8:

$P_{B1,3} = \{$**Specify data collection/storage procedures**$\}$		
Entry criteria:	-	$G' \neq \emptyset, E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset)$
Inputs:	-	$\{G', E', A, M, O\} \subset MO$
Tasks:	-	$\mathbf{P_{B1,3-1}} = \{$Identify existing sources of data that are generated from current work products, processes, or transactions.$\}$
	-	$\mathbf{P_{B1,3-2}} = \{$Identify measures for which data are needed, but are not currently available.$\}$
	-	$\mathbf{P_{B1,3-3}} = \{$Specify how to collect and store the data for each required measure.$\}$
	-	$\mathbf{P_{B1,3-4}} = \{$Create data collection mechanisms and process guidance for repsonsible personnel.$\}$
	-	$\mathbf{P_{B1,3-5}} = \{$Support automatic collection of the data where appropriate and feasible.$\}$
	-	$\mathbf{P_{B1,3-6}} = \{$Prioritize, review, and update data collection and storage procedures.$\}$
	-	$\mathbf{P_{B1,3-7}} = \{$Update measures and measurement goals as necessary.$\}$
Validation data:	-	The element count of the subset $D_{collection}$
	-	The element count of the subset $R_{collection}$
Outputs:	-	List of existing sources of data
	-	List of measures with unavailable data
	-	$D_{Collection} \subset D \in MO$
	-	$R_{Collection} \subset R \in MO$
E**X**it criteria:	-	$D_{Collection} \neq \emptyset$
	-	$R_{Collection} \neq \emptyset$

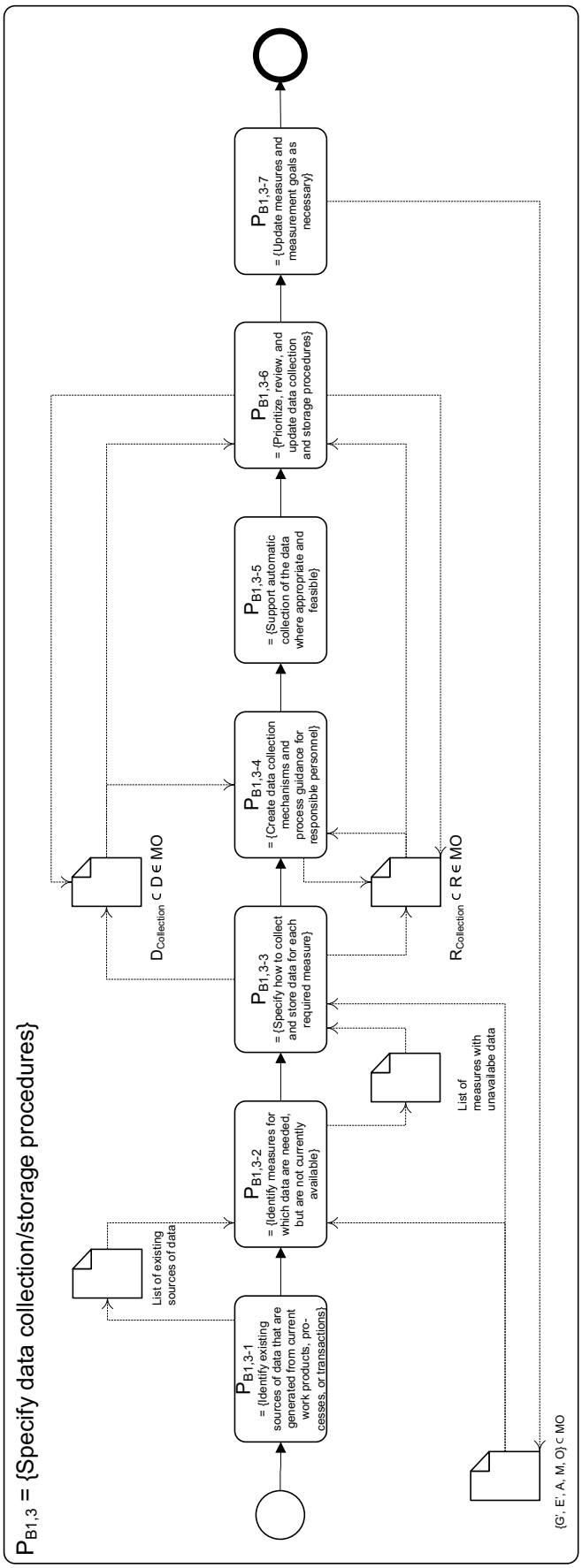

Figure 5.8: BPMN diagram of $P_{B1,3}$

The inner life of process $P_{B1,4}$

The final process step $P_{B1,4}$ in the process group P_{B1} deals with the specification of analysis procedures. In doing so, the first activity ($P_{B1,4-1}$) urges to specify and prioritize analyses that will be conducted and reports ($D_{Analysis}$) that will be prepared on the base of the up-till-then available information ($\{G', E', A, M, O\}$) of process step $P_{B1,2}$. Consequently, appropriate data analysis methods and tools can be selected ($P_{B1,4-2}$) and administrative procedures including the responsible personnel ($R_{Analysis}$) for analyzing and communicating the results specified ($P_{B1,4-3}$). After completion of the activities, the proposed contents and format of the specified analyses and reports can be updated ($P_{B1,4-4}$). In case measures and measurement goals were also subject to an alignment, they should be updated ($P_{B1,4-5}$). Finally, criteria ($D_{Analysis}$) for the evaluation of the analysis results and of the conduct of the measurement and analysis activities as utility should be specified ($P_{B1,4-3}$).

The following remarks according to the EITVOX process modeling paradigm and the BPMN diagram in figure 5.9 describe the activities, as well:

$P_{B1,4} = \{\textbf{Specify analysis procedures}\}$		
Entry criteria:	-	$G' \neq \emptyset, E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset$
	-	$D_{Collection} \neq \emptyset$
	-	$R_{Collection} \neq \emptyset$
Inputs:	-	$\{G', E', A, M, O\} \subset MO$
Tasks:	-	$\mathbf{P_{B1,4-1}} = \{$Specify and prioritize the analyses that will be conducted and the reports that will be prepared.$\}$
	-	$\mathbf{P_{B1,4-2}} = \{$Select appropriate data analysis methods and tools.$\}$
	-	$\mathbf{P_{B1,4-3}} = \{$Specify administrative procedures for analyzing the data and communicate the results.$\}$
	-	$\mathbf{P_{B1,4-4}} = \{$Review and update the proposed content and format of the specified analysies and reports.$\}$
	-	$\mathbf{P_{B1,4-5}} = \{$Update measures and measurement goals as neceassary.$\}$
	-	$\mathbf{P_{B1,4-6}} = \{$Specify criteria for evaluating the utility of the analysis results, and of the conduct of the measurement and analysis activities.$\}$
Validation data:	-	The element count of the subset $D_{Analysis}$
	-	The element count of the subset $R_{Analysis}$
Outputs:	-	$D_{Analysis} \subset D \in MO$
	-	$R_{Analysis} \subset R \in MO$
E**X**it criteria:	-	$D_{Analysis} \neq \emptyset$
	-	$R_{Analysis} \neq \emptyset$

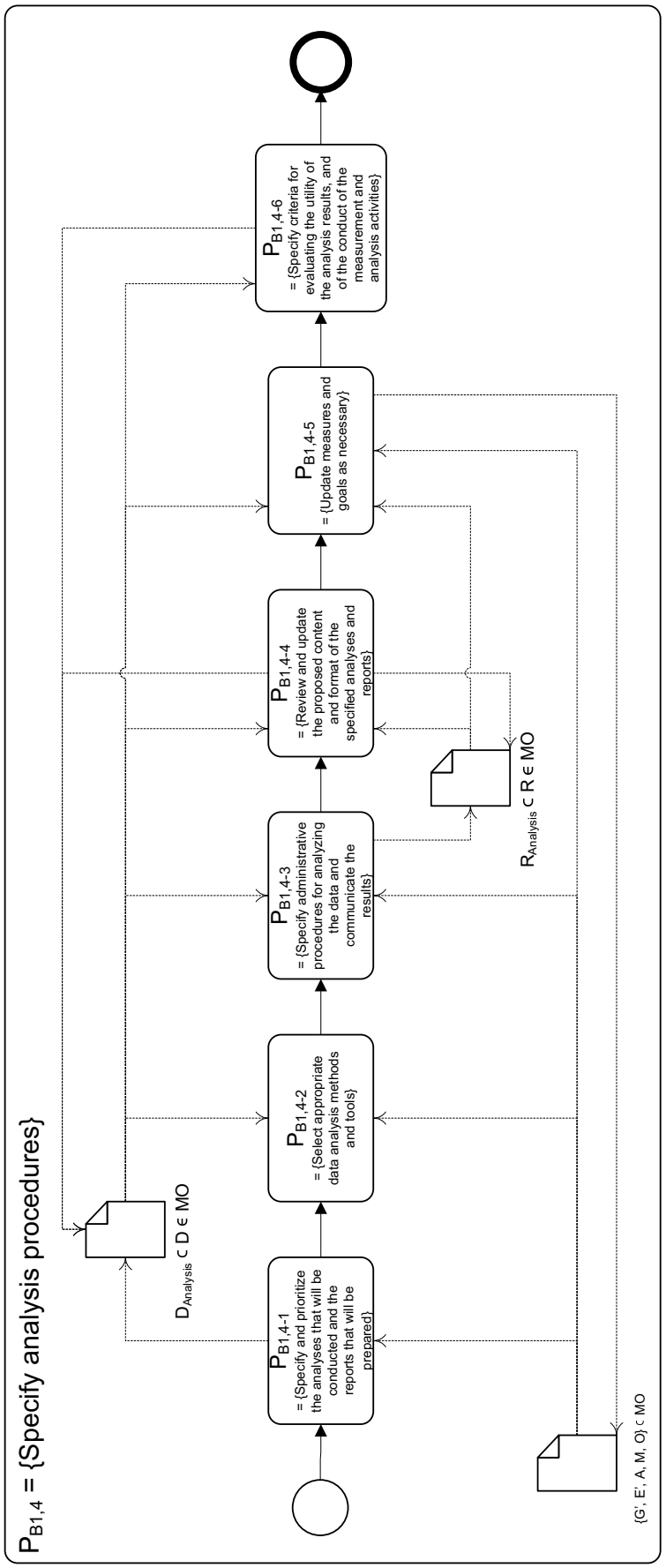

Figure 5.9: BPMN diagram of $P_{B1,4}$

5.4.2.2 Examining the process group P_{B2} at close quarters

Being in the convenient situation to be able to resort to a complete and available measurement outline (MO) that provides all the required information about established measurement goals (G'), selected software engineering entities of interest (E'), their attributes (A), operational definitions (O) as well as measurement directives (D) and the responsible personnel (R), the data collection process group can be started. Based on the measurement outline, measurement data (Q) can be collected ($P_{B2,1}$), and afterwards stored ($P_{B2,2}$) as formulated according to the EITVOX process modeling paradigm beneath and illustrated in figure 5.10:

$P_{B2} = \{Data\ collection\}$		
Entry criteria:	-	$G' \neq \emptyset; E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset; D \neq \emptyset; R \neq \emptyset$
Inputs:	-	$\{G', E', A, M, O, D, R\} \subseteq MO$
Tasks:	-	$\mathbf{P_{B2,1}} = \{\text{Collect and validate measurement data.}\}$
	-	$\mathbf{P_{B2,2}} = \{\text{Store measurement data.}\}$
Validation data:	-	The element count of the set Q
Outputs:	-	Q
E**X**it criteria:	-	$Q \neq \emptyset$

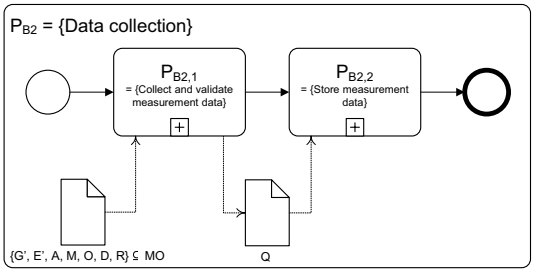

Figure 5.10: BPMN diagram of P_{B2}

The inner life of process $P_{B2,1}$
Taking the measurement outline (MO) as input, the base measures and/or quantities (Q_{Base}) can be obtained ($P_{B2,1-1}$), on the means of which the derived measures ($Q_{Derived}$) can be generated ($P_{B2,1-2}$), then. Finally, for both kinds of measures data integrity checks should be performed ($P_{B2,1-3}$) as close to the source of the data as possible. That context is also described in the table beneath and in figure 5.11:

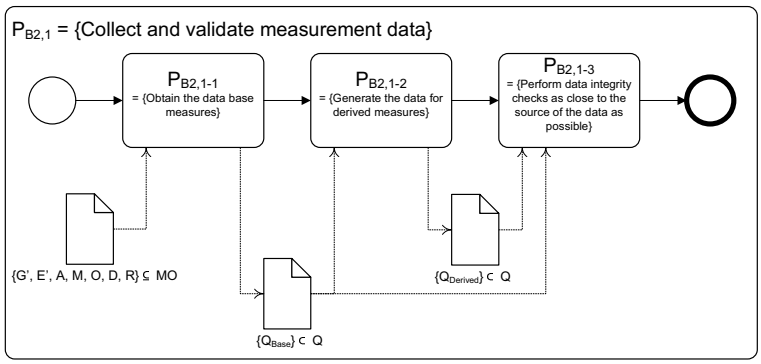

Figure 5.11: BPMN diagram of $P_{B2,1}$

$P_{B2,1} = \{\textbf{Collect and validate measurement data}\}$		
Entry criteria:	-	$G' \neq \emptyset; E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset; D \neq \emptyset; R \neq \emptyset$
Inputs:	-	$\{G', E', A, M, O, D, R\} \subseteq MO$
Tasks:	-	$\mathbf{P_{B2,1-1}} = \{$Obtain the data base measures.$\}$
	-	$\mathbf{P_{B2,1-2}} = \{$Generate the data for derived measures.$\}$
	-	$\mathbf{P_{B2,1-3}} = \{$Perform data integrity checks as close to the source of the data as possible.$\}$
Validation data:	-	The element count of the subsets Q_{Base} and $Q_{Derived}$
Outputs:	-	$Q_{Base} \subset Q$
	-	$Q_{Derived} \subset Q$
E**X**it criteria:	-	$Q_{Base} \neq \emptyset$
	-	$Q_{Derived} \neq \emptyset$

The inner life of process $P_{B2,2}$

Once more, taking the measurement outline (MO) as reference guideline, the measured quantities (Q) comprising base measures (Q_{Base}) and derived measures ($Q_{Derived}$) should be reviewed ($P_{B2,2-1}$) to ensure their completeness, integrity, accuracy, and currency. The stored quantities should only be made available ($P_{B2,2-2}$) for use by appropriate personnel and prevented ($P_{B2,2-3}$) from being used inappropriately. That context is also described in the table beneath and in figure 5.12:

$P_{B2,2} = \{\textbf{Store measurement data}\}$		
Entry criteria:	-	$G' \neq \emptyset; E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset; D \neq \emptyset; R \neq \emptyset; Q \neq \emptyset$
Inputs:	-	Q
	-	$\{G', E', A, M, O, D, R\} \subseteq MO$
Tasks:	-	$\mathbf{P_{B2,2-1}} = \{$Review the data to ensure their completeness, integrity, accuracy and currency.$\}$
	-	$\mathbf{P_{B2,2-2}} = \{$Make the stored contents available for use only by appropriate groups and personnel.$\}$
	-	$\mathbf{P_{B2,2-3}} = \{$Prevent the stored information from being used inappropriately.$\}$
Validation data:	-	None
Outputs:	-	None
E**X**it criteria:	-	None

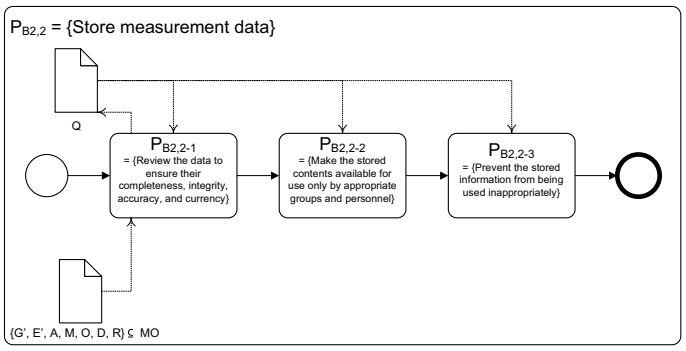

Figure 5.12: BPMN diagram of $P_{B2,2}$

5.4.2.3 Examining the process group P_{B3} at close quarters

After completion of collecting measurement data and/or quantities (Q) according to the measurement outline (MO) the quantities can be analyzed ($P_{B3,1}$), thereby producing measurement information products (I), which should be communicated ($P_{B3,2}$) as formulated according to the EITVOX process modeling paradigm beneath and illustrated in figure 5.13:

$P_{B3} = \{$**Data analysis**$\}$		
Entry criteria:	-	$G' \neq \emptyset; E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset; D \neq \emptyset; R \neq \emptyset$
	-	$Q \neq \emptyset$
Inputs:	-	$\{G', E', A, M, O, D, R\} \subseteq MO$
	-	Q
Tasks:	-	$\mathbf{P_{B3,1}} = \{\text{Analyze measurement data.}\}$
	-	$\mathbf{P_{B3,2}} = \{\text{Communicate results.}\}$
Validation data:	-	The element count of the set I
Outputs:	-	I
E**X**it criteria:	-	$I \neq \emptyset$

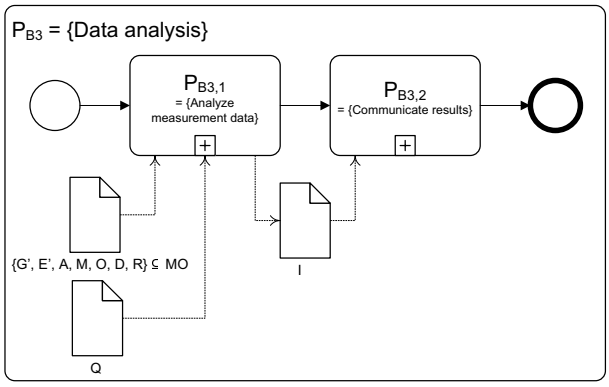

Figure 5.13: BPMN diagram of P_{B3}

The inner life of process $P_{B3,1}$
The measured base and derived quantities (Q) should be initially analyzed, interpreted, and conclusions should be drawn ($P_{B3,1-1}$) with respect to the information of the measurement outline (MO). This results in measurement information products (I), for which additional analysis should be performed and corresponding results prepared ($P_{B3,1-2}$). Because the information products are the potential foundation for decision making, the initial results should be reviewed together with the relevant stakeholders ($P_{B3,1-3}$) and analyses criteria should be refined for the future ($P_{B3,1-4}$). That context is also described in the table beneath and in figure 5.14:

$P_{B3,1}$ = {**Analyze measurement data**}		
Entry criteria:	-	$G' \neq \emptyset; E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset; D \neq \emptyset; R \neq \emptyset$
	-	$Q \neq \emptyset$
Inputs:	-	$\{G', E', A, M, O, D, R\} \subseteq MO$
	-	Q
Tasks:	-	$\mathbf{P_{3,B1-1}}$ = {Conduct initial analyses, interpret the results and draw preliminary conclusions.}
	-	$\mathbf{P_{3,B1-2}}$ = {Conduct additional measurement and analysis as necessary, and prepare results for interpretation.}
	-	$\mathbf{P_{3,B1-3}}$ = {Review the initial results with relevant stakeholders.}
	-	$\mathbf{P_{3,B1-4}}$ = {Refine criteria for future analyses.}
Validation data:	-	The element count of the set I
Outputs:	-	I
E**X**it criteria:	-	$I \neq \emptyset$

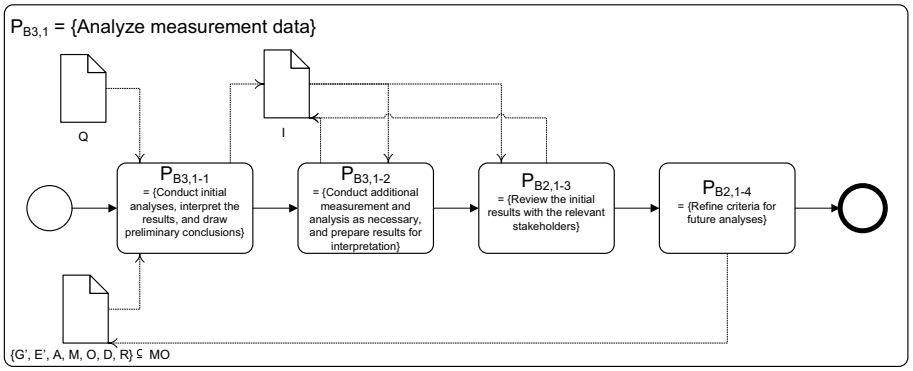

Figure 5.14: BPMN diagram of $P_{B3,1}$

The inner life of process $P_{B3,2}$

The final process $P_{B3,2}$ of the process group P_{B3} addresses the activities ($P_{B3,2-1}$) of keeping relevant stakeholders apprised of the measurement information products (I) on a timely basis as defined in parts of the measurement outline (MO) and the activities of assisting them in understanding the results ($P_{B3,2-2}$). That context is also described in the table beneath and in figure 5.15:

$P_{B3,2}$ = {**Communicate results**}		
Entry criteria:	-	$G' \neq \emptyset; E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset; D \neq \emptyset; R \neq \emptyset$
	-	$I \neq \emptyset$
Inputs:	-	$\{G', E', A, M, O, D, R\} \subseteq MO$
	-	I
Tasks:	-	$\mathbf{P_{B3,2-1}}$ = {Keep relevant stakeholders apprised of measurement results on a timely basis.}
	-	$\mathbf{P_{B3,2-2}}$ = {Assist relevant stakeholders in understanding the results.}
Validation data:	-	None
Outputs:	-	None
E**X**it criteria:	-	None

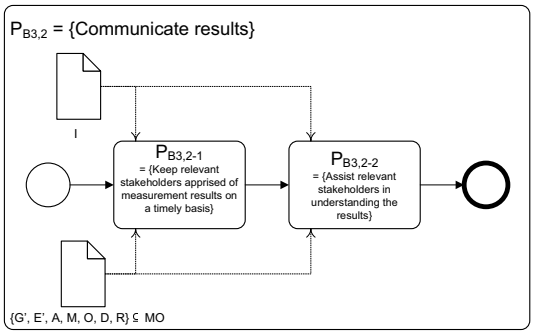

Figure 5.15: BPMN diagram of $P_{B3,2}$

5.4.2.4 Examining the process group P_{B4} at close quarters

Because the SMP has not been initiated with top-down defined measurement goals, but with proxy thresholds for the adopted software measures, causal analysis should be conducted. In combination with the information of the measurement outline (MO), variations of the measurement information products (I) from the defined thresholds should be appraised ($P_{B4,1}$) resulting in a list of revealed issues. Based on that list, pertinent root causes can be derived and translated to measurement goals (G''), which can then be used as input for the mixed (P_M) or top-down (P_T) sub-model of the SMPI model. The peculiarities are formulated according to the EITVOX process modeling paradigm beneath and illustrated in figure 5.16:

$P_{B4} = \{$**Causal analysis**$\}$		
Entry criteria:	-	$G' \neq \emptyset; E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset; D \neq \emptyset; R \neq \emptyset$
	-	$I \neq \emptyset$
Inputs:	-	$\{G', E', A, M, O, D, R\} \subseteq MO$
	-	I
Tasks:	-	$\mathbf{P_{B4,1}} = \{\text{Appraise variations from thresholds}\}$
	-	$\mathbf{P_{B4,2}} = \{\text{Translate root causes to measurement goals}\}$
Validation data:	-	The element count of the set G''
Outputs:	-	List of revealed issues
	-	G''
EX**i**t criteria:	-	$G'' \neq \{\emptyset\}$

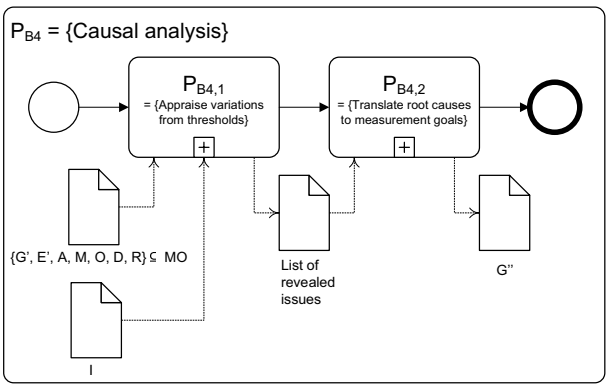

Figure 5.16: BPMN diagram of $P_{B,4}$

The inner life of process $P_{B4,1}$

In order to conduct an appraisal of variations from thresholds, the variations of the measurement information products (I) from the defined thresholds and/or proxy goals (G') should be identified ($P_{B4,1-1}$) as a list of variations. Having compiled such a list, the indicated issues and their impact should be examined concerning the degree of truth represented by the variations ($P_{B4,1-2}$). The resulting list of truly revealed issues should then be review, prioritized, and documented, accordingly ($P_{B4,1-3}$). That context is also described in the table beneath and in figure 5.17:

$P_{B4,1} =$ {**Appraise variations from thresholds**}		
Entry criteria:	-	$G' \neq \emptyset, I \neq \emptyset$
Inputs:	-	$\{G'\} \subset MO_{Bottom-up}$
	-	I
Tasks:	-	$\mathbf{P_{B4,1-1}} = \{$Identify variations of the measurement information products from the defined thresholds.$\}$
	-	$\mathbf{P_{B4,1-2}} = \{$Identify indicated issues and their impact by examining the degree of truth represented by these variations.$\}$
	-	$\mathbf{P_{B4,1-3}} = \{$Review, prioritize and select the revealed issues.$\}$
Validation data:	-	The element count of the set G''
Outputs:	-	List of variations
	-	List of revealed issues
	-	G''
E**X**it criteria:	-	$G'' \neq \emptyset$

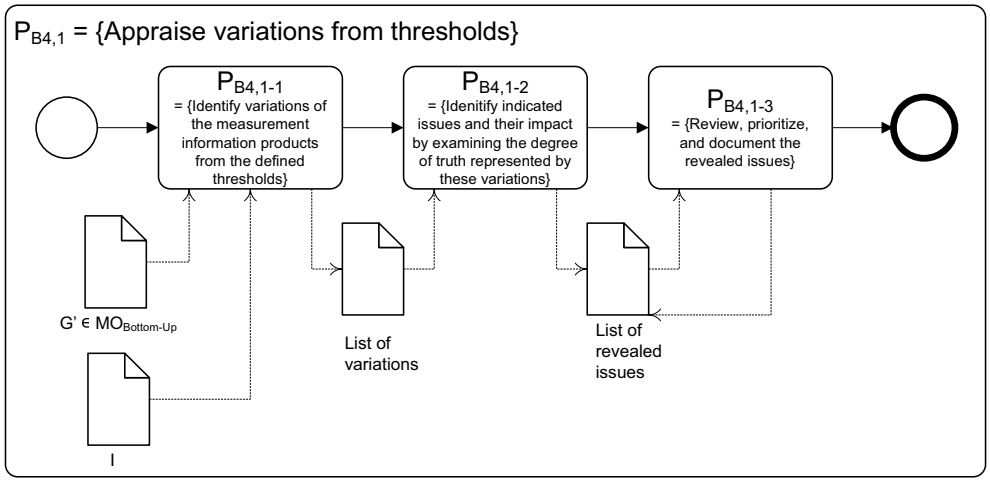

Figure 5.17: BPMN diagram of $P_{B4,1}$

The inner life of process $P_{B4,2}$

In order to translate root causes of variations from thresholds of selected software measures, the list of revealed issues should be analyzed and the reasons that underlay the documented issues should be identified until the root cause has been reached ($P_{B4,2-1}$). Then, from the produced list a number of root causes should be selected ($P_{B4,2-2}$) that are to be translated. Traversing the list of selected root causes, each of them should be rephrased ($P_{B4,2-3}$) to measurement goals (G'). Ultimately, these revealed measurement goals should be reviewed, prioritized, and documented ($P_{B4,2-4}$). That context is also described in the table beneath and in figure 5.18:

		$P_{B4,2} = \{$**Translate root causes to goals**$\}$
Entry criteria:	-	The list of revealed issues is complete and available.
Inputs:	-	The list of revealed issues
Tasks:	-	$\mathbf{P_{B4,2-1}} = \{$Identify the reasons underlying the documented issues until the root cause has been reached.$\}$
	-	$\mathbf{P_{B4,2-2}} = \{$Select a number of root causes to be translated.$\}$
	-	$\mathbf{P_{B4,2-3}} = \{$Rephrase the root causes to measurement goals.$\}$
	-	$\mathbf{P_{B4,2-4}} = \{$Review, prioritize, and document the revealed measurement goals.$\}$
Validation data:	-	The element count of the set G''
Outputs:	-	List of root causes
	-	List of selected root coot causes
	-	G''
EXit criteria:	-	$G'' \neq \emptyset$

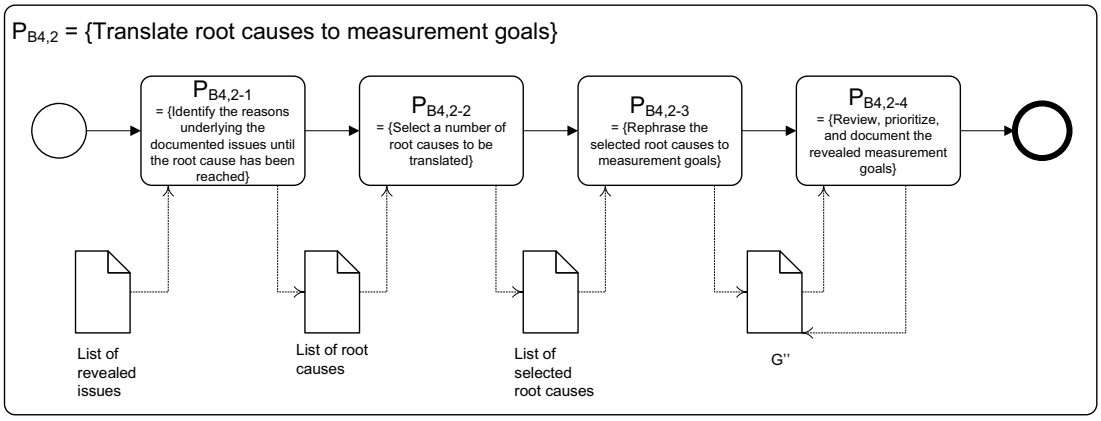

Figure 5.18: BPMN diagram of $P_{B4,2}$

5.4.3 P_M — The mixed sub-model in detail

The mixed sub-model (P_M) as illustrated in figure 5.19 would then fit best for an interested organization, if there was a successful bottom-up software measurement program in place, which should be aligned with a set of defined measurement goals to drive the software measurement process. In connection with a properly staffed, well-funded measurement project this is a sign of basic software measurement experiences, on which more sophisticated software measurement processes towards the holistic goal-orientation can be implemented.

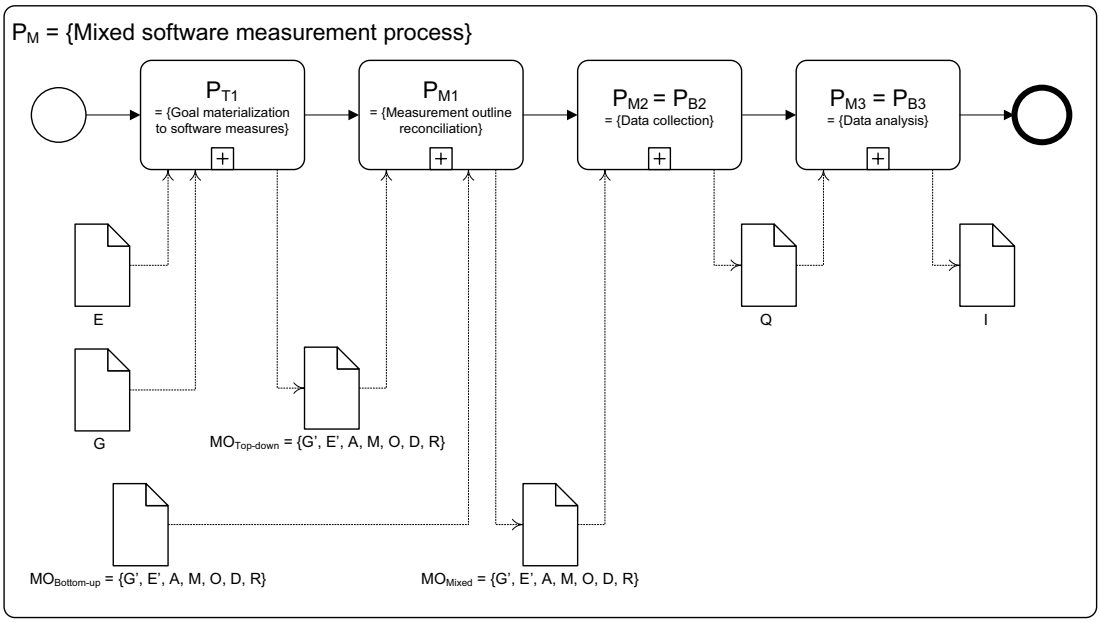

Figure 5.19: BPMN diagram of P_M

The first step in the mixed model is the materialization (P_{T1}) of measurement goals (G) that have been defined in a top-down manner under usage of software engineering entities (E). In the next step, the resulting top-down measurement outline ($MO_{Top-down}$) is reconciliated (P_{M1}) with the bottom-up measurement outline ($MO_{Bottom-up}$) of a previous measurement process conduct following the bottom-up measurement paradigm. The mixed measurement outline (MO_{Mixed}), which results from the reconciliation process, is then used for data collection (P_{M2}) of measured quantities (Q), which are used for data analysis (P_{M3}) to derive measurement information products (I), in turn. As becomes clear from figure 5.19, because the process of 'goal materialization to software measures' (P_{T1}) is identical to the one defined in the next subsection and the data collection and data analysis processes do not differ from the ones of the bottom-up sub-model (P_B) of the SMPI model, they are referenced ($P_{M2} = P_{B2}, P_{M3} = P_{B3}$) but not modeled again for reasons of space.

$P_{Mixed} = P_M$		
Entry criteria:	-	$G \neq \emptyset, E \neq \emptyset, MO_{Bottom-up}$ is complete and available
Inputs:	-	G
	-	E
	-	$MO_{Bottom-up}$
Tasks:	-	$\mathbf{P_{T1}} = \{Goal\ materialization\ to\ software\ measures\}$
	-	$\mathbf{P_{M1}} = \{Measurement\ outline\ reconciliation\}$
	-	$\mathbf{P_{M2}} = \{Data\ collection\} = P_{B2}$
	-	$\mathbf{P_{M3}} = \{Data\ analysis\} = P_{B3}$
	-	$\mathbf{P_{M4}} = \emptyset$
Validation data:	-	The element counts of the subsets of MO_{Mixed}
	-	The element count of the sets Q, I
Outputs:	-	$MO_{Top-down}$
	-	MO_{Mixed}
	-	Q
	-	I
E**X**it criteria:	-	MO_{Mixed} is complete and available, $Q \neq \emptyset, I \neq \emptyset$

5.4.3.1 Examining the process group P_{M1} at close quarters

The process group P_{M1} of measurement outline reconciliation avails itself of the information of the top-down measurement outline ($MO_{Top-down}$) as defined within the preceding process group and the bottom-up measurement outline ($MO_{Bottom-up}$) of a previous measurement process run. The first process ($P_{M1,1}$) consolidates both measurement outlines resulting in reconciliated established measurement goals (G'), software engineering entities of interest (E), their measured attributes (A), software measures (M), and operational definitions (O) as part of the mixed measurement outline (MO_{Mixed}). Adjacently, the data collection and storage procedures and the analysis procedures and/or directives (D) together with the respectively responsible personnel (R) are aligned ($P_{M1,2}$)($P_{M1,3}$) with these reconciliated sets as formulated according to the EITVOX process modeling paradigm beneath and illustrated in figure 5.20:

$P_{M1} = \{\mathbf{Measurement\ outline\ reconciliation}\}$		
Entry criteria:	-	$MO_{Top-down}$ is complete and available
	-	$MO_{Bottom-up}$ is complete and available
Inputs:	-	$MO_{Top-down}$
	-	$MO_{Bottom-up}$
Tasks:	-	$\mathbf{P_{M1,1}} = \{\text{Consolidate measurement outlines.}\}$
	-	$\mathbf{P_{M1,2}} = \{\text{Consolidate data collection procedures.}\}$
	-	$\mathbf{P_{M1,3}} = \{\text{Consolidate analysis procedures.}\}$
Validation data:	-	The element count of the subsets of MO_{Mixed}
Outputs:	-	MO_{Mixed}
E**X**it criteria:	-	MO_{Mixed} is complete and available

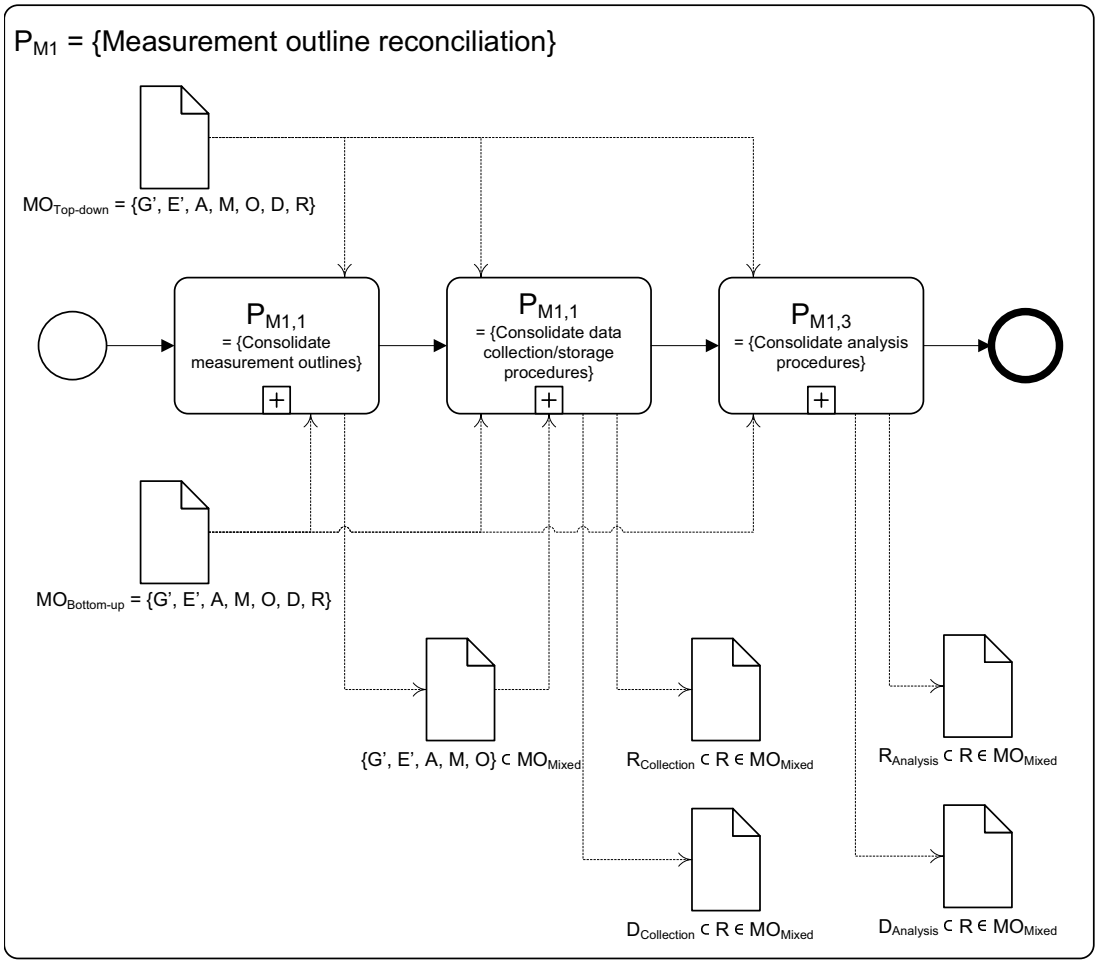

Figure 5.20: BPMN diagram of P_{M1}

The inner life of process $P_{M1,1}$

To consolidate the measurement outlines towards a mixed measurement outline (MO_{Mixed}), software measures ($M_{Bottom-up} \subset M_{Mixed}$) from the bottom-up measurement outline ($MO_{Bottom-up}$) that are of avail for the ones of the goal-oriented measurement outline ($MO_{Top-down}$) have to be identified ($P_{M1,1-1}$). Together with these measures the goal-oriented measurement outline is examined to identify ($P_{M1,1-2}$) additional software measures to be added ($M_{Top-down} \subset M_{Mixed}$). After having completed the set of mixed software measures (M_{Mixed}) the resulting mixed measurement outline is to be replenished ($P_{M1,1-3}$) with information concerning software engineering entities (E) to quantify, their attributes (A), and operational definitions (O). In fact, there is the reconciliated set of mixed software measures, but the measurement goals have not been reconciliated and documented yet. This is to be catched up by tracing back ($P_{M1,1-4}$) the software measures from the mixed measurement outline to measurement goals (G'). Afterwards, the results of the process ($\{G', E', A, M, O\} \subset MO_{Mixed}$) should be reviewed as well as documented ($P_{M1,1-5}$) and communicated to all involved stakeholders ($P_{M1,1-6}$). That context is also described in the table beneath and in figure 5.21:

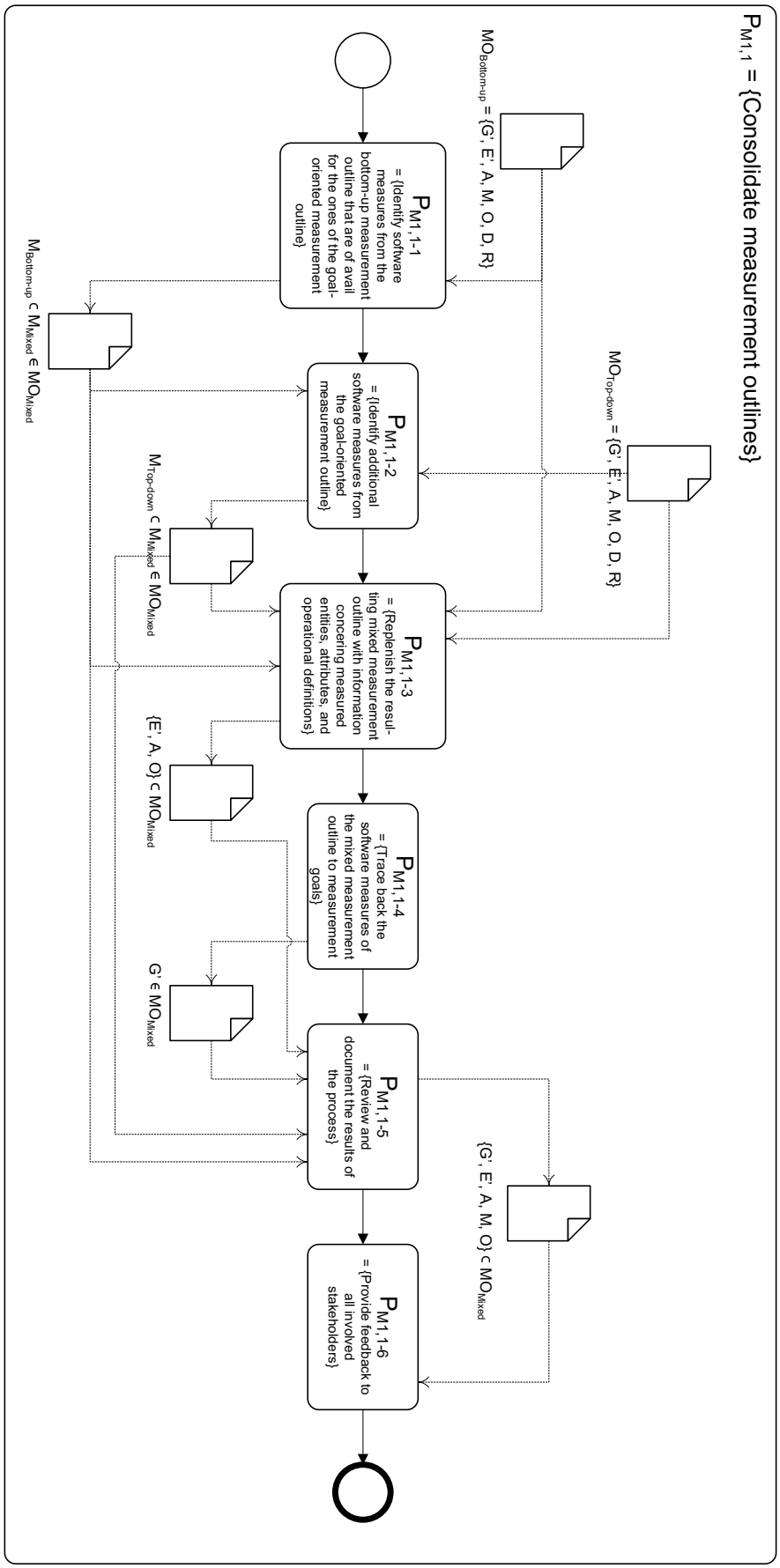

Figure 5.21: BPMN diagram of $P_{M1,1}$

$P_{M1,1} = \{\textbf{Consolidate measurement outlines}\}$		
Entry criteria:	-	$MO_{Top-down}, MO_{Bottom-up}$ are complete and available
Inputs:	-	$MO_{Top-down}$
	-	$MO_{Bottom-up}$
Tasks:	-	$\mathbf{P_{M1,1-1}} = \{$Identify software measures from the bottom − up measurement outline that are of avail for the ones of the goal − oriented measurement outline.$\}$
	-	$\mathbf{P_{M1,1-2}} = \{$Identify additional software measures from the goal − oriented measurement outline.$\}$
	-	$\mathbf{P_{M1,1-3}} = \{$Replenish the resulting mixed measurement outline with information concerning measured entities, attributes, and operational definitions.$\}$
	-	$\mathbf{P_{M1,1-4}} = \{$Trace back the software measures of the mixed measurement outline to measurement goals.$\}$
	-	$\mathbf{P_{M1,1-5}} = \{$Review and document the results of the process.$\}$
	-	$\mathbf{P_{M1,1-6}} = \{$Provide feedback to all involved stakeholders.$\}$
Validation data:	-	The element count of the subsets $\{G', E', A, M, O\} \subset MO_{Mixed}$
Outputs:	-	$\{G', E', A, M, O\} \subset MO_{Mixed}$
E**X**it criteria:	-	$G' \neq \emptyset; E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset$

The inner life of process $P_{M1,2}$
The consolidation of data collection and storage procedures follows the consolidation of the measurement outlines. On the base of both measurement outlines, the bottom-up ($MO_{Bottom-up}$) and the top-down ($MO_{Top-down}$) one, existing data collection and storage procedures ($D_{Collection_{Bottom-up}}, D_{Collection_{Top-down}}$) as well as the responsible personnel ($R_{Collection_{Bottom-up}}, R_{Collection_{Top-down}}$) for the software measures comprised in the mixed measurement outline (MO_{Mixed}) should be identified ($P_{M1,2-1}$). Then, the mixed measurement outline (MO_{Mixed}) should be replenished with these information ($P_{M1,2-2}$) and afterwards reviewed and corrected ($P_{M1,2-3}$) as necessary. That context is also described in the table beneath and in figure 5.22:

$P_{M1,2} = \{$**Consolidate data collection/storage procedures**.$\}$		
Entry criteria:	-	$MO_{Top-down}, MO_{Bottom-up}$ are complete and available
Inputs:	-	$MO_{Top-down}$
	-	$MO_{Bottom-up}$
Tasks:	-	$\mathbf{P_{M1,2-1}} = \{$Identify existing data collection/storage procedures and responsible personnel for the software measures comprised in the mixed measurement outline.$\}$
	-	$\mathbf{P_{M1,2-2}} = \{$Replenish the mixed measurement outline with information concerning data collection/storage procedures and responsible personnel.$\}$
	-	$\mathbf{P_{M1,2-3}} = \{$Review and correct the results of the process as neceassary.$\}$
Validation data:	-	The element count of the subsets $D_{Collection} \subset D \in MO_{Mixed}$
	-	The element count of the subsets $R_{Collection} \subset R \in MO_{Mixed}$
Outputs:	-	$D_{Collection} \subset D \in MO_{Mixed}$
	-	$R_{Collection} \subset R \in MO_{Mixed}$
E**X**it criteria:	-	$D_{Collection} \neq \emptyset, R_{Collection} \neq \emptyset$

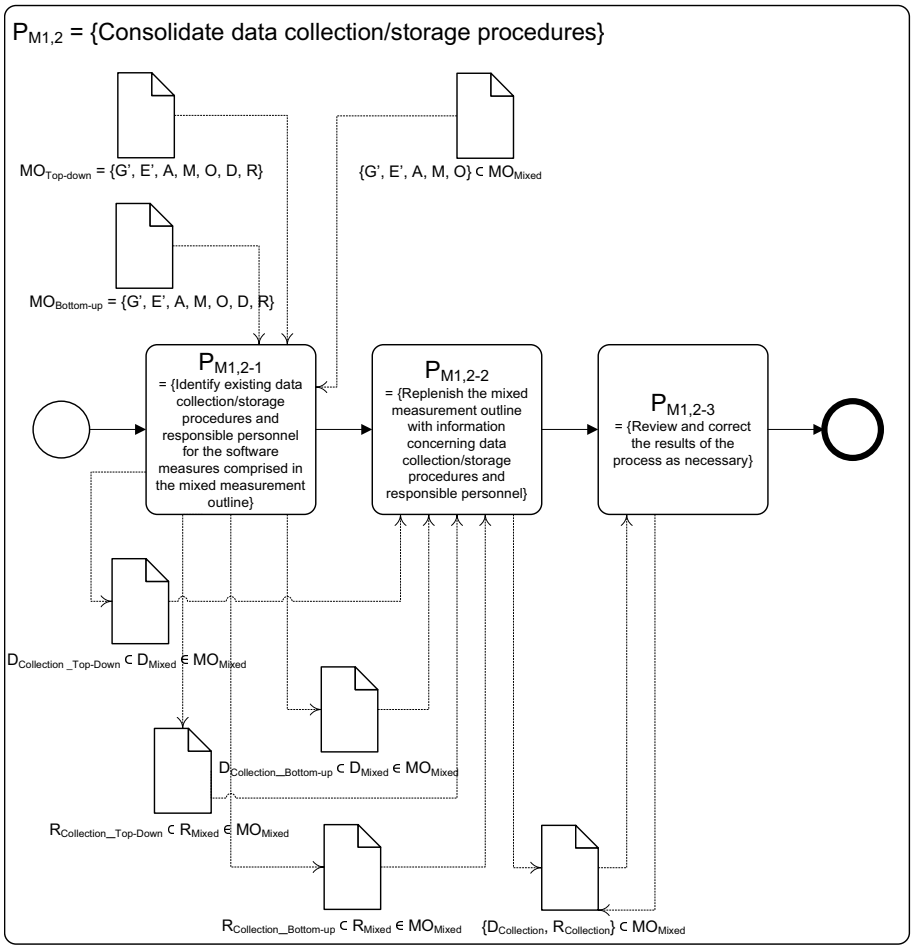

Figure 5.22: BPMN diagram of $P_{M1,2}$

The final process of reconciliation between a bottom-up measurement outline ($MO_{Bottom-up}$) and a top-down one ($MO_{Top-down}$) deals with the consolidation of analysis procedures. Therefore, existing analysis procedures ($D_{Analysis_{Bottom-up}}$, $D_{Analysis_{Top-down}}$) and responsible personnel ($R_{Analysis_{Bottom-up}}, R_{Analysis_{Top-down}}$) for the measures comprised in the mixed measurement outline (MO_{Mixed}) should be identified ($P_{M1,3-1}$). Once, the information are available and the activity is completed, the mixed measurement outline should be replenished ($P_{M1,3-2}$) with these information as well as reviewed and corrected ($P_{M1,3-3}$), if necessary. That context is also described in the table beneath and in figure 5.23:

$P_{M1,3} = \{$**Consolidate data collection/storage procedures**.$\}$		
Entry criteria:	-	$MO_{Top-down}, MO_{Bottom-up}$ are complete and available
Inputs:	-	$MO_{Top-down}$
	-	$MO_{Bottom-up}$
Tasks:	-	$\mathbf{P_{M1,3-1}} = \{$Identify existing analysis procedures and responsible personnel for the software measures comprised in the mixed measurement outline.$\}$
	-	$\mathbf{P_{M1,3-2}} = \{$Replenish the mixed measurement outline with information concerning data analysis procedures and responsible personnel.$\}$
	-	$\mathbf{P_{M1,3-3}} = \{$Review and correct the results of the process as neceassary.$\}$
Validation data:	-	The element count of the subsets $D_{Analysis} \subset D \in MO_{Mixed}$
	-	The element count of the subsets $R_{Analysis} \subset R \in MO_{Mixed}$
Outputs:	-	$D_{Analysis} \subset D \in MO_{Mixed}$
	-	$R_{Analysis} \subset R \in MO_{Mixed}$
E**X**it criteria:	-	$D_{Analysis} \neq \emptyset, R_{Analysis} \neq \emptyset$

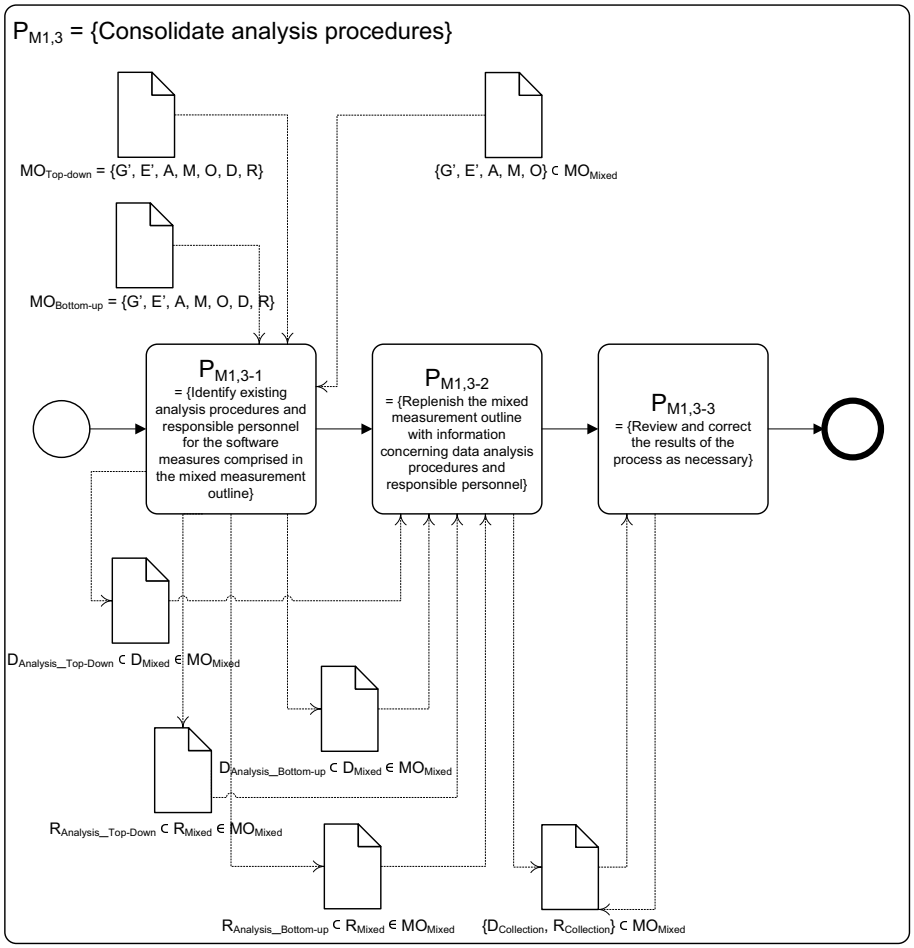

Figure 5.23: BPMN diagram of $P_{M1,3}$

5.4.4 P_T — The top-down sub-model in detail

After all, at the top scale of the paradigm-specific sub-models of the SMPI model, the top-down sub-model (P_T) resides. Apart from a perfectly set up measurement project, software measurement at this stage of process improvement is only conducted according to important business goals. There are no data collection activities that end up in data cemeteries. This pinpoints on a huge amount of experience with software measurement processes. By following the top-down sub-model these processes can be stabilized for later improvement of their capability according to the 'Measurement & Analysis' support process area being part of the continuous representation of SEI's CMMI.

Analog to the mixed sub-model (P_M), the materialization of measurement goals (G) that have been defined in a top-down manner under usage of software engineering entities (E) forms the first step (P_{T1}). The resulting top-down measurement outline ($MO_{Top-down}$) is then taken as input for collecting (P_{T2}) measurement data and/or quantities (Q), which are analyzed (P_{T3}) in order to gain measurement information products (I). As becomes clear from figure 5.24, because the data collection and data analysis processes do not differ from the ones of the bottom-up sub-model (P_B) of the SMPI model, they are referenced ($P_{T2} = P_{B2}, P_{T3} = P_{B3}$) but not modeled again for reasons of space, once more.

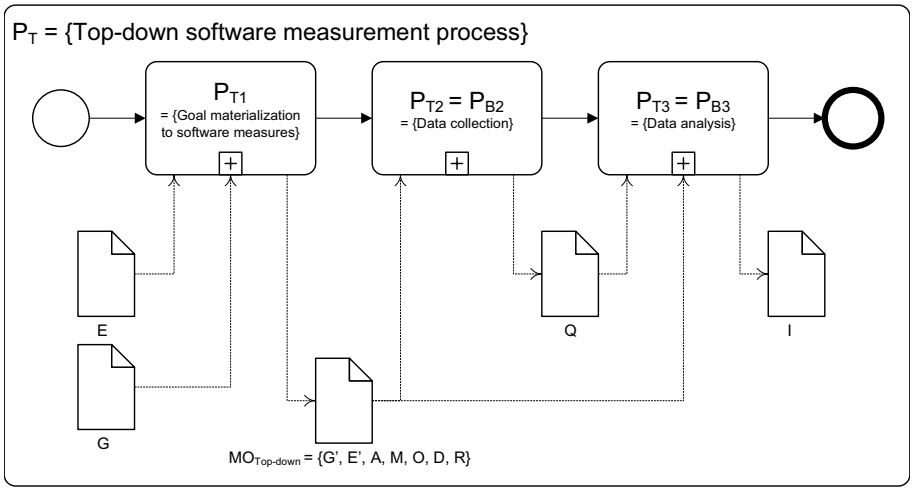

Figure 5.24: BPMN diagram of P_T

$P_{Top-down} = P_T$		
Entry criteria:	-	$G \neq \emptyset, E \neq \emptyset$
Inputs:	-	G
	-	E
Tasks:	-	$\mathbf{P_{T1}} = \{\text{Goal materialization to software measures}\}$
	-	$\mathbf{P_{T2}} = \{\text{Data collection}\} = P_{B2}$
	-	$\mathbf{P_{T3}} = \{\text{Data analysis}\} = P_{B3}$
	-	$\mathbf{P_{T4}} = \{\emptyset\}$
Validation data:	-	The element counts of the subsets of $MO_{Top-down}$
	-	The element count of the sets Q, I
Outputs:	-	$MO_{Top-down}$
	-	Q
	-	I
E**X**it criteria:	-	$MO_{Top-down}$ is complete and available
	-	$Q \neq \emptyset, I \neq \emptyset$

5.4.4.1 Examining the process group P_{T1} at close quarters

Starting from a set of general measurement goals (G), the first process addresses the establishment ($P_{T1,1}$) of specific measurement objectives and/or goals (G'). Together with the set of software engineering entities of an organization, the specification of measures ($P_{T1,2}$) yields to the selection of software engineering entities of interest (E'), their attributes (A), software measures (M), and operational definitions (O). Based on these information, procedures for data collection/storage ($D_{Collection}$) and analysis ($D_{Analysis}$) as well as the respectively responsible personnel ($R_{Collection}, R_{Analysis}$) shall be produced ($P_{T1,3}, P_{T1,4}$). Taken altogether, a top-down measurement outline ($MO_{Top-down}$)

As depicted in figure 5.25 the processes of specifying data collection/storage procedures and specifying analysis procedures are identical to the ones defined in the bottom-up model (P_B) of the SMPI model, it is, once more, forbeared from presenting it with redundancy. In lieu thereof, the processes are solely referenced (($P_{T1,3} = P_{B1,3}, P_{T1,4} = P_{B1,4}$)).

$P_{T1} = \{\textbf{Goal materialization to software measures}\}$		
Entry criteria:	-	$G \neq \emptyset, E \neq \emptyset$
Inputs:	-	G
	-	E
Tasks:	-	$\mathbf{P_{T1,1}} = \{\text{Establish measurement objectives}\}$
	-	$\mathbf{P_{T1,2}} = \{\text{Specify measures}\}$
	-	$\mathbf{P_{T1,3}} = \{\text{Specify data collection/storage procedures}\} = P_{B1,3}$
	-	$\mathbf{P_{T1,4}} = \{\text{Specify analysis procedures}\} = P_{B1,4}$
Validation data:	-	The element counts of the subsets of $MO_{Top-down}$
Outputs:	-	$MO_{Top-down}$
E**X**it criteria:	-	$MO_{Top-down}$ is complete and available

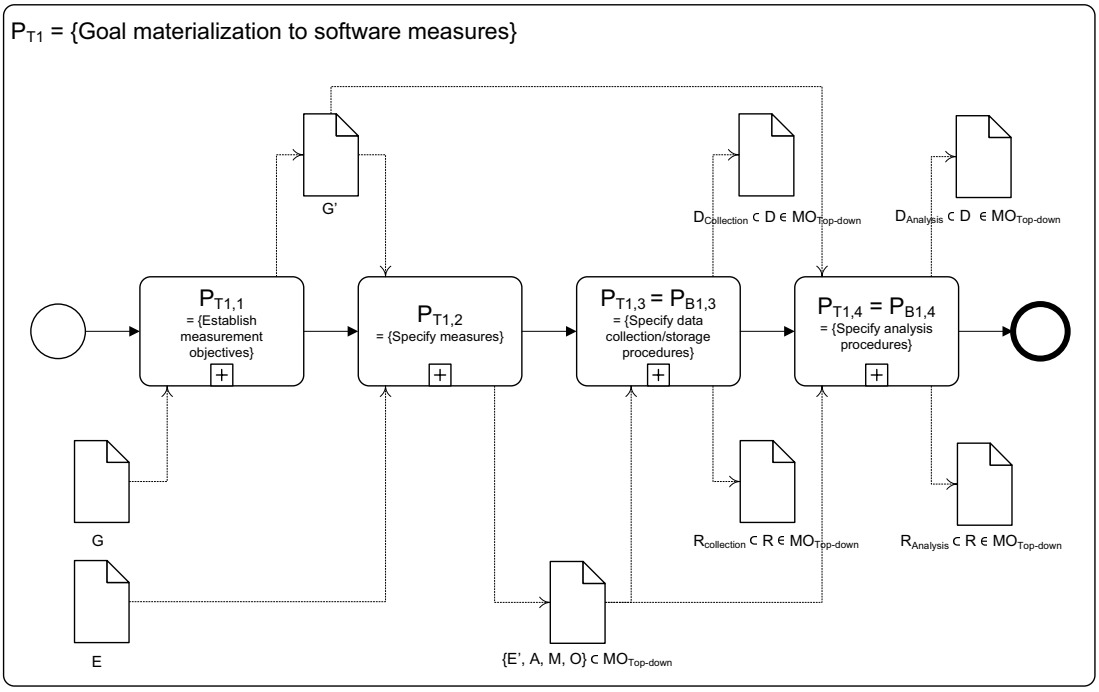

Figure 5.25: BPMN diagram of P_{T1}

The inner life of process $P_{T1,1}$
In order to establish specific measurement objectives and/or goals, the set of general measurement goals (G) is taken as input for the documentation ($P_{T1,1-1}$) of information needs and objectives. As a result the records on the list of information needs and objectives should be prioritized ($P_{T1,1-2}$) yielding to a prioritized list. In turn, that list should be taken to document, review, and update ($P_{T1,1-3}$) the specific measurement objectives and/or goals (G'). Then, feedback for refining and clarifying the information needs and objectives as necessary, should be given ($P_{T1,1-4}$) and traceability of the measurement objectives to the identified information needs and objectives should be maintained ($P_{T1,1-5}$). That context is also described in the table beneath and illustrated in figure 5.26:

$P_{T1,1} = \{\textbf{Establish measurement objectives}\}$		
Entry criteria:	-	$G \neq \emptyset$
Inputs:	-	G
Tasks:	-	$\mathbf{P_{T1,1-1}} = \{\text{Document information needs and objectives.}\}$
	-	$\mathbf{P_{T1,1-2}} = \{\text{Prioritize information needs and objectives.}\}$
	-	$\mathbf{P_{T1,1-3}} = \{\text{Document, review, and update measurement objectives.}\}$
	-	$\mathbf{P_{T1,1-4}} = \{\text{Provide feedback for refining and clarifying information needs and objectives as necessary.}\}$
	-	$\mathbf{P_{T1,1-5}} = \{\text{Maintain traceability of the measurement objectives to the identified information needs and objectives.}\}$
Validation data:	-	The element count of the set G'
Outputs:	-	List of information needs and objectives
	-	List of prioritized information needs and objectives
	-	$G' \in MO_{Top-down}$
E**X**it criteria:	-	$G' \neq \emptyset$

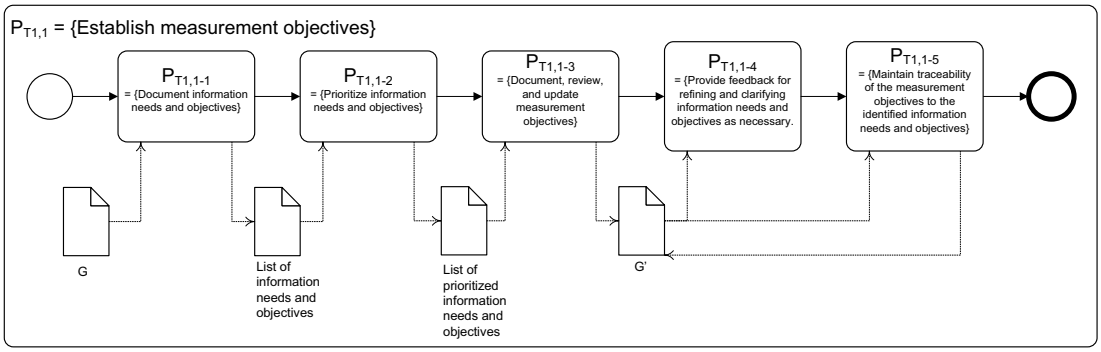

Figure 5.26: BPMN diagram of $P_{T1,1}$

The inner life of process $P_{T1,2}$
Accompanied by the revealed set of established, specific measurement goals (G') and the general set of software engineering entities (E) of an organization, candidate software measures should be identified ($P_{T1,2-1}$) with respect to the documented specific measurement goals. This results in a set of selected software engineering entities (E'), their attributes to be quantified (A) by identified software measures (M). Next, existing measures that already address the specific measurement goals should be identified ($P_{T1,2-2}$) yielding to a list of existing measures. With the aid of that list, operational definitions (O) can be either adopted from existing software measures or specified from scratch for new ones ($P_{T1,2-3}$). After all, the measures should be prioritized, reviewed, documented, and if necessary updated ($P_{T1,2-4}$). That context is also described in the table beneath and illustrated in figure 5.27:

$P_{T1,2} = \{\textbf{Specify measures}\}$		
Entry criteria:	-	$G' \neq \emptyset, E \neq \emptyset$
Inputs:	-	G'
	-	E
Tasks:	-	$\mathbf{P_{T1,2-1}} = \{$Identify candidate measures based on documented$\}$ measurement goals.$\}$
	-	$\mathbf{P_{T1,2-2}} = \{$Identify existing measures that already address$\}$ the measurement goals.$\}$
	-	$\mathbf{P_{T1,2-3}} = \{$Specify operational definitions for the measures.$\}$
	-	$\mathbf{P_{T1,2-4}} = \{$Prioritize, review, document and update measures.$\}$
Validation data:	-	The element count of the subsets $\{E', A, M, O\}$.
Outputs:	-	$\{E', A, M, O\} \subset MO_{Top-down}$
E**X**it criteria:	-	$E' \neq \emptyset; A \neq \emptyset; M \neq \emptyset; O \neq \emptyset$

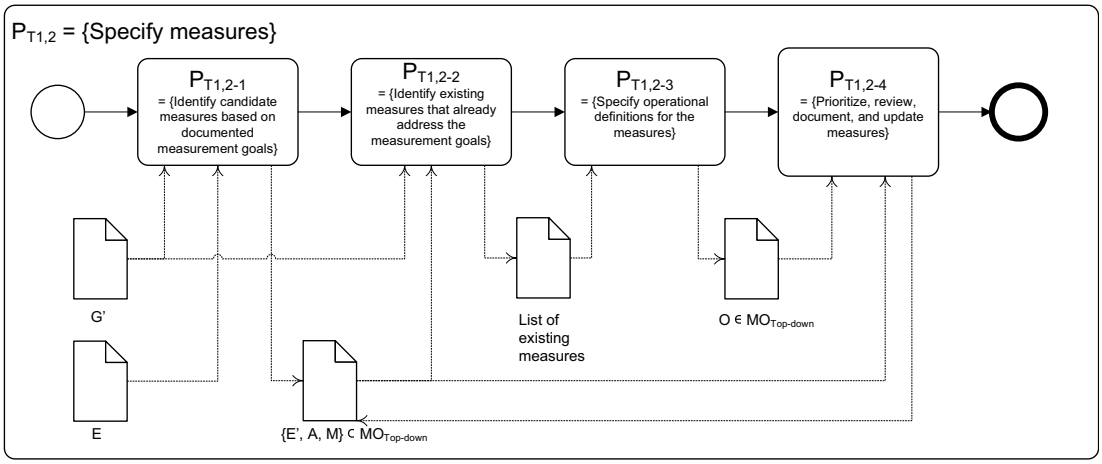

Figure 5.27: BPMN diagram of $P_{T1,2}$

5.5 Conclusion

By means of the chapter at hand, the author of this thesis addressed the fourth and fifth sub-questions of this research project:

RQ4. What would be the concept to overcome the specific shortcomings of the process improvement model finally extracted in RQ3 with respect to the criteria of RQ1/RQ2?

RQ5. How can the approach proposed as result of RQ4 be transferred into a stepwise software measurement process improvement model?

As revealed in chapter 4 the CMMI-MA support process area of the CMMI Framework v1.1 exemplarily implements the related, up-to-date standards, thereby fully complying with the software measurement process model of ISO/IEC Standard 15939 and

with the regulations of ISO/IEC Standard 15504 SPICE, roughly speaking. Thus, it was accepted as a basis model for software measurement process improvement. Recognizing its shortcoming of dealing with the top-down software measurement paradigm alone and disregarding the important bottom-up and mixed approaches, the model of the 'Measurement & Analysis' support process area was selected as intellectual foundation. In order to overcome its revealed shortcoming, the design and development concept (DC) was to develop a supplemental process model covering the improvement of the software measurement process by running through all of the measurement paradigms, one after another. At the same time a jump into a paradigm-specific submodel relative to the level of ability to define measurement goals should be possible, too.

In order to answer the sixth sub-question of this research work, three development rationales (DR1, DR2, DR3) were set up: A consensus of phases of the software measurement process as extracted from reference literature and specific for each of the software measurement paradigms, was produced. This provided a framework for the process model of the complemented SMPI model and its paradigm-related sub-models. Next, the distinct phases of the process sub-models were filled with life. That is, applicable contents was mapped from reference literature and best practices as mentioned in chapter 2, where possible. In the cases where the reference literature kept quiet, the development of the contents was pursued independently by the author. Starting from the CMMI-MA as basis model, the in the described manner complemented model forms the empirical, descriptive SMPI model seen from the functional, behavioral, organizational, and informational perspectives for the management domain (cf. section 3.2). After all, the SMPI model as product of the author's scientific work was presented both, the whole model at a glance and each of the paradigm-specific sub-models in detail. In doing so, the graphical diagrams using the BPMN of OMG was complemented by a textual formulation using the EITVOX process modeling paradigm, which is very common in industry and should ease the adoption of the model in the target group in industry.

When reducing the SMPI model to the basic building blocks, a process system taxonomy different from the one presented in chapter 3 by Wang et al. [WK00] was chosen to provide a contrast. A mapping between the taxonomy of Wang et al. and the virtual process elements of the model can be easily conducted as shown in table 5.4*.

Process system taxonomy [WK00]	**Process elements of the SMPI model**
Process system	*SMPI model* (Count: 1) (P)
Process subsystem	*Paradigm-specific sub-models* (Count: 3) (P_B, P_M, P_T)
Process category	*Process groups* (Count: 10) (P_{Ba}, P_{Mh}, P_{Tx})
Process	*Processes* (Count: 15) ($P_{Ba,b}, P_{Mh,i}, P_{Tx,y}$)
Practice	*Activities* (Count: 65) ($P_{Ba,b-c}, P_{Mh,i-j}, P_{Tx,y-z}$)

Table 5.4: Confrontation of the general process system taxonomy with the process elements of the SMPI model

*The count of distinct process elements of the SMPI model model is provided in brackets.

Ultimately, the features of the SMPI model as adopted from the CMMI-MA and developed in this chapter, shall be recapitulated again:

	RQ1 (Content-related criteria)	*CMMI-MA*	*SMPI*
C1.	The **scope** of the process improvement model must be (at least partially) on software measurement process implementation in industrial settings.	✓	✓
C2.	The process improvement model must have been developed with **scientific rigor**.	✓	✓
C3.	The process improvement model must be able to reflect the sequential application of the 'bottom-up', 'mixed', and 'top-down' **measurement paradigms** during implementation of software measurement processes in industrial settings.	–	✓
	RQ2 (Model-related criteria)	*CMMI-MA*	*SMPI*
M1.	The process improvement model must provide a **process reference model** compliant with the requirements of ISO/IEC Standard 15504.	✓	✓
M2.	The process improvement model must be **compatible with the measurement framework** of ISO/IEC Standard 15504.	✓	✓
M3.	The process improvement model's **process assessment model** must represent a mapping of the process reference model and the measurement framework of ISO/IEC Standard 15504.	✓	✓
M4.	The process improvement model's **assessment process** must observe the regulations of ISO/IEC Standard 15504.	✓	✓
M5.	The process improvement model must provide process **improvement guidelines** for the specified scope.	✓	✓
	RQ3 (Basis model and its shortcomings)		
B1.	'Measurement & Analysis' (CMMI-MA) support process area of the CMMI Framework v1.1		*CMMI-MA*

B2.	Lack of support of the sequential application of the 'bottom-up', 'mixed', and 'top-down' **measurement paradigms** during implementation of the software measurement process.	*CMMI-MA*	
	RQ4 (Development concept)	*CMMI-MA*	*SMPI*
DC.	Extend the 'Measurement & Analysis' support process area of the CMMI Framework v1.1 so that it can support the sequential application of the 'bottom-up', 'mixed', and 'top-down' **measurement paradigms** during implementation of the software measurement process in industrial settings!	–	✓
	RQ5 (Development rationale)	*CMMI-MA*	*SMPI*
DR1.	Establish consensus among the phases of the different software measurement process models specific for each measurement paradigm.	–	✓
DR2.	Perform a mapping of processes and activities onto the consensus phases of the different software measurement process models specific for each paradigm. Bring life into topics not covered.	–	✓
DR3.	Provide a graphical and a textual formulation for each of the different software measurement process models specific for each measurement paradigm.	–	✓

After the completion of the development work as presented in this chapter, the validation of the SMPI model in the target area of industry application has to follow. Two different models, that is the ones of CMMI-MA and of SMPI, are available. Thus, a statistical test of two different hypotheses has to be performed: The null-hypothesis (H_0) and the alternative hypothesis (H_1), which can be defined as following:

	RQ6 (Case-study validation using statistical test of hypotheses)
H_0	*Validating appropriately, there is no difference in improving the implementation and sustainment of the software measurement process for an unexperienced organization between the usage of the CMMI-MA model and the SMPI model.*
H_1	*Validating appropriately, the SMPI model better supports the implementation and sustainment of the software measurement process for an unexperienced organization than the CMMI-MA.*

Part III

Measure, analyze, evaluate

Chapter 6

Work Validation

"Case studies help industry evaluate the benefits of methods and tools and provide a cost-effective way to ensure that process changes provide the desired results."

– Barbara A. Kitchenham, Lesley Pickard, and Shari Lawrence Pfleeger [KPF95] –*

6.1 Introduction

Undoubtedly, the stepwise SMPI process model for improving the implementation and sustainment of the software measurement process in industrial settings should be tested, whether a real improvement compared with the effects of a traditional approach to the problem might be attested for industrial software engineering settings. Only if this was the case, the improvement approach would be useful and the research project could be successfully completed.

As mentioned earlier (cf. 1.2.5) there are several ways to validate methods or models in empirical software engineering such as surveys, experiments, and case studies, with each of them depending on the scope and/or extent of the validation efforts. The author counts himself lucky for having been given the opportunity to prove the case for the SMPI model in a single project in an industrial software engineering organization as a validation procedure that is commonly called case study or pilot study. Fortunately, there is some good advice available on how to perform experimental research in software engineering in general, and on how to perform case studies as 'research in the typical' in particular. [FPG94] [KPP95] [Kit96a] [Gla97] [KPP$^+$02] And it was felt that adherence to those good advices is more than necessary for a case study itself to cope with the four criteria of research-design quality (construct, internal, external validity, and experimental reliability) Yin [Yin02, p. 34] postulates. To ensure research-design quality, the seven-step 'Case Study Guidelines' proposed by Kitchenham et al. [KPP95] [Kit96a] were adopted and brought together with their later remarks [KPP$^+$02] concerning the experimental context, experimental design, the conduct of the study and data collection, analysis, presentation of results, and interpretation of results. Furthermore, because a successful application has been multiply reported [BC91] [BMA96] [ALB99] [Apr05] elements of Basili's framework [BSH86, Bas93] were interspersed, too. In the following, the conducted case study will be described successively according to the key points.

*English researchers and practitioners in software engineering, luminaries of software measurement

6.2 The case study's industrial context

Because describing the circumstances, under which the empirical case study evaluation was conducted, can aid interested organizations in deciding, whether the improvement approach is viable to be replicated in the own environment, it is described next. A precondition to conduct the empirical investigation in the host organization was the author's commitment to sign and respect a non-disclosure agreement. In it, the organization, being different from the author's current employer, obligates him to withhold any information that allows one to draw conclusions on the organization's brand name, its projects, or products, because it was considered strategically important. However, the anonymity achieved by sanitizing the information should not pose a problem or affect the general validity of the results.

After having been separated from the parent group and bought up, the host organization, being situated in the southern part of Germany, purely operated in the software engineering sector. Starting from a significantly higher but not communicated number of software developers, who developed a wider range of software products before the separation, merely fifty of them bearing a good deal of experience remained in the organization, at the time of the case study. Then, just exclusively developing application software in the trenches of an evolutionary, incremental development process, the company navigated in choppy water. However, the top management of the asset stripper triggered the introduction of a SPI initiative based on the SEI's CMMI Framework v1.1 in the organization. Having set the target profile of process areas required in maturity level two of the model's staged representation, the preparation of process descriptions was already completed. In contrast, the achievements in other process areas were proceeding unassertively or simply failed. A failure was especially the case for the support process area 'Measurement & Analysis', because software measurement had been neglected ever before.

Thus, a predisposition to try out the improvement approach of the author's SMPI model to implement and sustain a SMP from scratch in the course of a case study was in place, backed up by funding and commitment of the host organization's management. Due to the narrow schedule of a remaining preparation phase for an internal Standard CMMI Appraisal Method for Process Improvement (SCAMPI) class C assessment of one year, the sponsoring management of the host organization set the timescale for the case study to three months at maximum. Therefore, a special 'Measurement & Analysis' project was set up having adequate funding and unprejudiced personnel consisting of one employee full-time and two people working half-time and on other projects and having been assigned by normal staff-allocation procedures. Because all of the project team members were not experienced in leading a project, a role of the project leader was appointed arbitrarily to one of them.

6.3 Discussion of hypotheses

The lack of experience with the software measurement process within the small project team paired with the urgent need to show off results, set the stage for testing both approaches independently. That is, the traditional top-down software measurement process model as manifested in the CMMI Framework's 'Measurement & Analysis' support process area as well as the SMPI model that additionally provides processes following the bottom-up and mixed measurement paradigms on top. Having the top management's implied engagement to test the traditional approach to SMP implementation and sustainment at first as treatment one and in case of failure, the improved approach

of the author as treatment two, the null-hypothesis (H_0) and the alternative hypothesis (H_1) to be tested were formulated as following:

> H_0 *There is no difference in improving the implementation and sustainment of the software measurement process for an unexperienced organization between the usage of the CMMI-MA model and the SMPI model.*
>
> H_1 *The SMPI model better supports the implementation and sustainment of the software measurement process for an unexperienced organization than the CMMI-MA.*

6.4 Case study design

The single project case study approach yields to the restriction of its population to a single organization with the primary unit of analysis being the unparalleled project of implementing and sustaining a SMP for pilot project producing application software X. Because the author, as researcher planning and organizing the performance of the case study, has had only a small stake in the build-up of the project team for the 'Measurement & Analysis' project of the host organization, no sampling procedure could be used. However, the concerns of the author to take across the team of the previously failed 'Measurement & Analysis' project led to the allocation of three unprejudiced staff members having the same lack of measurement experience as any other employee of the organization. So it was hoped that the effects of learning from mistakes or elevated enthusiasm and/or skepticism of the team members would not interfere the case study.

Because there was vested interest of the author to prove the case for the SMPI model, apart from an introduction of the constituents of both treatments, the project staff should be left alone with the process descriptions to reduce bias. Moreover, the author should solely be in the position of an observer. The general outcomes of the case study were then defined by the management of the company relatively fuzzy:

- "Implement a working software measurement program with respect to the 'Measurement & Analysis' support process area of the CMMI Framework v1.1 within three months."
- "Sustain the implemented measurement program at least until the SCAMPI class C assessment in approximately one year."

These guidelines have been translated into the following outcome measures for the end of the case study (1) and/or a revisit after the SCAMPI class C assessment (2):

1. The existence of a set of top-down defined *measurement goals* (G), a resulting *measurement outline* (MO), accordingly *measured quantities* (Q) as well as the *measurement information products* (I) satisfying the top-down defined measurement goals.
2. The existence of (1) at the time of the SCAMPI class C assessment.

6.5 Conduct of the case study and data collection

Owing to the considered strategic importance, the non disclosure agreement with the host organization forbids the unvarnished publication of intermediate work products. Thus, the description of the conduct of the case study, is restricted to rather summarized events. However, the author feels that this does not hinder the description's significance.

6.5.1 Application of treatment one

At the beginning, the project team members familiarized themselves with the 'Measurement & Analysis' support process area of the CMMI Framework v1.1, once more. When they recognized that a set of measurement goals was required as input to the measurement process, management was asked to provide their measurement goals. But because in management only a fuzzy need for improvement prevailed that was rather caused by gut-feel than by being able to assess the current position, no measurement information needs and/or goals could be provided. After two weeks of intensive discussions, the project team capitulated with the application of treatment one. This decision was backed up by hindsight of management that goal-oriented measurement without goals was a useless undertaking and resembled willful waste of resources.

Thus the application of treatment one for that organization and/or measurement project with management and staff being completely unexperienced in software measurement in general and goal-setting in particular, was suspended without any outcome after two calendar weeks and approximately 20 person days of effort of the measurement project team.

6.5.2 Application of treatment two

The application of treatment two, that is the author's SMPI model, was started at the beginning of the fourth calendar week by the same project members. In consequence of the early failure of treatment one, the improved model was studied. Because the inability of management to set solid measurement goals did not die away, the improved process model was given a chance because it helps in setting these goals by running through all of its sub-models, successively. A sketch of events that occurred during the execution of the process groups of each sub-model is given beneath.

1. Execution of the bottom-up sub-model of the SMPI model
P_{B1} — *Selection of software measures:* Based on a small literature study and on an inquiry of a German professional association dealing with software measurement in general, a universe of software measures was selected. After informal discussions with management, the number of software measures was restricted to only one, that was quality of software X in terms of the number of post-release defects after one month of usage, as similarly proposed by Putnam and Myers [PM03]. For that software measure, a threshold was set by discussing with other software developers and compliance with that threshold was set as proxy measurement goal. Fostered by the completed, parallel and stumbled in introduction of a commercial-off-the-shelf defect data base, the data collection and storage procedures for the number of post-release defects for a product within the one-month post-release period were provided as part of the technology introduction project of the company. For this simple software measure the analysis obligations were easily provided.

P_{B2} — *Data collection:* Benefiting from the parallel introduction of the defect data base and the obligation for software developers to track incoming defects according to the process defined by the technology introduction project, data collection and storage were performed exemplarily.

P_{B3} — *Data analysis:* The analysis of the measured quantities with respect to the set measurement goal or rather threshold, yielded to a disclosure of strong variances which were communicated to management.

P_{B4} — *Causal analysis:* After having been pointed to a variation from the set threshold for the quality-related software measure, management appraised the variation and regarded this to be of importance. During a closed meeting, management identified the root cause and rephrased it into the first top-down defined measurement goals: (1) Identify failure-prone components of software X prior to release. (2) Identify defects of software X in the first month after release.

The execution of the bottom-up sub-model of the SMPI model as a first step could be completed after six calendar weeks (approximately 60 person days of effort of the measurement project team) resulting in an intermediate results (measured quantities (Q), measurement information products (I), a measurement outline ($MO_{Bottom-up}$), and a set of measurement goals (G) derived by causal analysis) that satisfied all stakeholders, at least for the moment.

2. Execution of the mixed sub-model of the SMPI model

P_{T1} — *Goal materialization to software measures:* The two measurement goals defined as the result of the execution of the bottom-up sub-model of the SMPI model, were reckoned as mandatory. Accordingly, two software measures (size in terms of lines of code, and complexity in terms of McCabe's cyclomatic complexity [McC76]) were defined for goal (1) and one (the already established defect count for the first month after release) for goal (2). Especially for the two software measures, required for the quantification of goal (1) the definition of data collection/storage and analysis procedures, which would fit to the defined operational definitions of the software measures, was tedious but could be successfully completed. To promote the later collection of the software measures for goal (1), a small parser was provided as plug-in module to the integrated development environment used by the software developers.

P_{M1} — *Measurement outline reconciliation:* Having the opportunity to resort on the measurement outline of the bottom-up sub-model of the SMPI model, the reconciliation between the newly defined mixed measurement outline and the previous one was relatively easy. Because the only one software measure of the bottom-up measurement outline was still of interest and thus also comprised in the recent measurement outline, in this special case the measures as well as the data collection/storage and analysis procedures were not subject to reconciliation.

P_{M2} — *Data collection:* While the data collection for the defect count rapidly became a routine activity of the software developers, data collection for the chosen size and complexity software measures was burdensome despite of the provided plug-in tool. However, face-to-face promotion of these software measures and the measurement goals behind them yielded to appreciation among software developers and support — some times grudgingly.

P_{M3} — *Data analysis:* The result of analyzing the measured quantities towards measurement information products was twofold: On the one hand side, the analysis of the software measures used for the quantification of goal (1) pointed on weak spots in the development process and also on failure-prone components of software X. On the other hand side, the ongoing analysis of the defect count for the first month after release of software X did not yield to an improved situation of the faults that occurred at the user sites as defects. But the team of the 'Measurement & Analysis' project

understood that this was not due. Moreover, the similar observations of the bottom-up sub-model were used to improve goal-setting that resulted in the exposition of the weak points, which could only then be addressed because they have been made explicit. Communicating the results to management, appropriate corrective action was planned and taken.

The conduct of mixed sub-model of the SMPI as the second step of software measurement process improvement could be completed after a period of eight weeks (approximately 80 person days of effort of the measurement project team), what meant not meeting the case study's timescale by three weeks. However, since the measurement project was prosperous and management had confidence, this did not turn out to be a major problem but unavoidable point of critique. The prosperity of the 'Measurement & Analysis' project of the host company was mirrored by an increased number of measured quantities (Q), a mixed measurement outline (MO_{Mixed}) , and the provision of measurement information products (I) that led to corrective actions. But more important from the vantage point of software measurement process improvement, was the apparently emerged ability of management to set measurement goals (G) as result of the mixed sub-model based on the results of the bottom-up sub-model of the SMPI model.

At the closure of the case study, the mixed sub-model was run through successfully and the preconditions for conducting a top-down software measurement process as required by the 'Measurement & Analysis' support process area of the CMMI Framework v1.1 were established for the first time. All stakeholders expressed their contentedness and relaxation.

3. Execution of the mixed sub-model of the SMPI model
Now, that some months have gone by, since the conduct of the case study, a follow-up telephone call with the leader (and today process owner) of the 'Measurement & Analysis' project at the host organization attested the continued prosperity and usefulness of the now top-down oriented software measurement process and/or SMP. Also the SCAMPI class C assessment gave evidence of the conformity of the SMP with the 'Measurement & Analysis' support process area of the CMMI Framework v1.1. Thus, all expectations of the host organization's management could be obviously satisfied.

6.6 Result interpretation and conclusion

This chapter dealt with the last sub-question of this research project:

> *RQ6. What is the result of a case study validation in industry of the stepwise software measurement process improvement model resulting from RQ5?*

Interpreting the results of the above mentioned case study, the following can be summarized for the given industrial context: First, treatment one, that is the application of the traditional top-down measurement process model as manifested in the CMMI-MA support process area, failed and did not produce any utilizable outcome measure. Second, all the outcome measures for treatment two, that is the application

of the SMPI model that provides measurement process improvement along the bottom-up and mixed measurement paradigms on top of the process model of treatment one, were positively satisfied.

With respect to the hypotheses as stated in section 6.3, the author comes to the logical conclusion that there are very well differences between the two approaches. More specifically, the author's improved approach of the SMPI model better supports implementation and sustainment of a SMP for an unexperienced organization than the process model of the 'Measurement & Analysis' support process area of the CMMI Framework v1.1. Thus, the quintessence is:

Reject H_0 in favor of H_1!

Owing to the fact that, the proposed SMPI model gave evidence of its validity in practice, the research project is regarded to have been successfully completed.

Chapter 7

Summary

"Discovery consists of seeing what everybody has seen and thinking what nobody has thought."

– Albert Szent-György[*] –

The thesis at hand presents the steps taken and the results of a PhD research project to answer the following main research objective:

> *How can the implementation and sustainment of the software measurement process in industrial settings be improved?*

In 2004, at the time of initially being confronted with this task from process engineering, the author was aware of the fact that this research objective must have been addressed before with the results not preventing the question from being asked, undeviatingly. Thus, the engineering research paradigm was chosen to tackle the research issue thereby trying to adopt the advantageous features of prior attempts and/or software measurement process improvement models, find their weak points, and overcome or improve them. The research was then subdivided into three distinct parts:

Part I — Observation of existing solutions: In order to be able to evaluate prior attempts to the problem, evaluation criteria had to be set up. Similarly to semantical and syntactical properties of source code, content-related and model-related criteria for existing software measurement process improvement models had to be found. By means of the second chapter the most important aspects having an impact on the implementation and sustainment of the software measurement process in software engineering industry have been elicited and positioned in a whole picture. Based on that, the content-related evaluation criteria for potential solutions to the research objective were proposed. Afterwards in chapter three, and based on the theory of software process engineering, model-related evaluation criteria were extracted from a review of characteristics of mainstream SPA/SPI models and of figures of merit set by the ISO/IEC Standard 15504. Once the criteria were fully explored and documented, existing software measurement process improvement models were reviewed and assessed against these criteria from chapter four. The result of this evaluation was that the 'Measurement & Analysis' (CMMI-MA) support process area of the SEI's CMMI Framework v1.1 enjoyed supremacy when compared with other models. Because of that it was adopted as basis model for this research. However, it shipwrecked when it came to support software measurement process improvement along the measurement paradigms as revealed in chapter two. This shortcoming provided the opportunity to improve the model in the second part of the research project.

[*]Hungarian physiologist, winner of the 1937 Nobel Prize in medicine, and former professor at the University of Budapest, Hungary, *1893 – †1986

Part II — Proposal and development of a better solution: Having exposed the omission of the selected basis solution, CMMI-MA, in terms of the support for software measurement processes following paradigms being less challenging than the top-down one, a development concept for a complemented model could be formulated. Because it intends to support software measurement process improvement along all the measurement paradigms, this complemented model was then coined the *Software Measurement Process Improvement (SMPI)* model. Subsequently, the conceived design and development rationale envisaged a consensus mapping of available reference material concerning phases and contents of the process sub-models specific to each distinct measurement paradigm. Moreover, to ease adaptation in industrial software engineering settings the SMPI model was presented graphically with the aid of diagrams compliant with the BPMN of OMG and textually using the EITVOX notation.

Part III — Measure, analyze, evaluate: Although being based on an up-to-date, encyclopedical inventory of software measurement best practices, the developed SMPI model had to prove its applicability in an external case study in practice. In addition to that, a progress in terms of a better capability to improving the implementation of the software measurement process in an industrial software engineering setting had to be substantiated. A host organization in need of implementing software measurement processes, which is situated in the southern part of Germany, agreed to evaluate the SMPI model and could prove the model's advancement in software measurement process improvement capabilities, when compared with the CMMI-MA support process area.

7.1 Main contributions

The principal part of this thesis makes the following main contributions to research within the field of software measurement in the discipline of software engineering:

(1) An up-to-date, encyclopedical inventory of best practices and aspects to be observed for implementation and sustainment of the software measurement process in industrial settings.

(2) The set of specific, content-related and generally-applicable, model-related criteria to be fulfilled by software measurement process improvement models dwelling on its implementation and sustainment.

(3) The evaluation of existing implicit or explicit software measurement process improvement models according to the criteria mentioned above.

(4) The empirical, descriptive *Software Measurement Process Improvement (SMPI)* model that stepwise facilitates the improvement of the software measurement process along the measurement paradigms. This can be a significant aid in implementing and sustaining SMPs.

(5) The presentation of an industrial case study corroborating the validity of the developed SMPI model.

The secondary products of this thesis, which can be found in the appendix, are:

(A1) The clear and brief presentation of measurement theory applicable to software engineering settings.

(A2) A glimpse of mainstream models for SPA/SPI in software engineering.

7.2 Future work

As already mentioned in the introductory part of this thesis, the engineering research cycle was run through once to assure theoretical correctness at first and later the applicability and usefulness in practice. From present appreciation can be guessed that due to further advancements in theory and practice of software measurement process implementation and sustainment minor factors of the model might be subject to change but major aspects will remain universally valid.

However, it is felt that the success of the developed SMPI model should be validated in a broader field. This could be done in a diversified empirical investigation spanning on different companies of different sizes and orientations. By following a piggyback manner to process improvement endeavors the users that still were doubtfully of the model's advantages could be convinced. With the aid of further empirical studies, the complex interrelations could also be investigated further and the model could be enhanced and/or optimized, accordingly.

Together with the author, mentoring researchers came to the recommendation that the developed and validated SMPI model should be proposed for being considered in a future revision of the SEI's CMMI Framework, and more specifically in its 'Measurement & Analysis' support process area.

Bibliography

[Aae03] Ivan Aaen. Software process improvement: Blueprints versus recipes. *IEEE Software*, 20(5):86–93, 2003.

[AAJS99] Erik Arisholm, Bente Anda, Magna Jørgensen, and Dag I. K. Sjøberg. Guidelines on conducting software process improvement studies in industry. In *Proceedings of the 22nd IRIS Conference (Information Systems Research Seminar In Scandinavia)*. Department of Informatics, University of Oslo, Norway, 1999.

[Abr01] Pekka Abrahamsson. Commitment development in software process improvement: critical misconceptions. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 71–80, Washington, DC, USA, 2001. IEEE Computer Society.

[aC94] SEI at CMU. Course notebook for defining software processes. a process improvement course offered by the SEI, 1994. Pittsburgh, PA, USA.

[AC95] Alain April and François Coallier. Trillium: a model for the assessment of telecom software system development and maintenance capability. In *ISESS '95: Proceedings of the 2nd IEEE Software Engineering Standards Symposium*, pages 175–183, Washington, DC, USA, August 1995. IEEE Computer Society.

[ACF97] Vincenzo Ambriola, Reidar Conradi, and Alfonso Fuggetta. Assessing process-centered software engineering environments. *ACM Transactions on Software Engineering Methodology (TOSEM)*, 6(3):283–328, July 1997.

[ACT03] Dennis M. Ahern, Aaron Clouse, and Richard Turner. *CMMI Distilled: A Practical Introduction to Integrated Process Improvement*. Addison-Wesley Professional, Boston, MA, USA, 2nd edition, September 2003. ISBN: 0321186133.

[AD99] Alain Abran and Reiner R. Dumke, editors. *Proceedings of the 9th International Workshop on Software Measurement IWSM*, Montreal, QC, Canada, September 8–10 1999.

[Adr93] W. Richards Adrion. Research methodology in software engineering: Summary of the dagstuhl workshop on future directions in software engineering. *ACM SIGSoft Software Engineering Notes*, 18(1):36–37, January 1993.

[AG83] Allan J. Albrecht and John E. Gaffney. Software function, source lines of code, and development effort prediction: A software science validation. *IEEE Transactions on Software Engineering*, SE-9(6):639–648, November 1983.

[Alb79] Allan J. Albrecht. Measuring applications development productivity. In *Proceedings of IBM Application Development Joint SHARE/GUIDE Symposium*, pages 83–92, Monterey, CA, USA, 1979. IBM.

[ALB99] Alain Abran, Lucie Laframboise, and Pierre Bourque. A risk assessment method and grid for software measurement programs. Technical Report 99-03, Département d'informatique, Université du Québec à Montréal, Montréal, QC, Canada, 1999.

[ALMN99] David E. Avison, Francis Lau, Michael D. Myers, and Peter Axel Nielsen. Action research. *Communications of the ACM*, 42(1):94–97, 1999.

[Alp87] Theodore M. Alper. A classification of all order-preserving homeomorphism groups of the reals that satisfy finite uniqueness. *Journal of Mathematical Psychology*, 31(2):135–154, 1987.

[AM06] Kiumi Akingbehin and Bruce Maxim. A three-layer model for software engineering metrics. In *Proceedings of the Seventh ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD'06)*, pages 17–20, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[AMBD04] Alain Abran, James W. Moore, Pierre Bourque, and Robert Dupuis, editors. *SWEBOK — Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, Los Alamitos, CA, USA, February 2004.

[Apr05] Alain April. *S3m — Model to Evaluate and Improve the Quality of Software Maintenance Process*. PhD thesis, Department of Computer Science, University of Magdeburg, Germany, November 2005.

[Ari01] Erik Arisholm. *Empirical Assessment of Changeability in Object-Oriented Software*. PhD thesis, Department of Informatics, Faculty of Mathematics and Natural Sciences, University of Oslo, Oslo, Norway, February 2001.

[AS91] Chris Argyris and Donald A. Schön. *Participatory Action Research*, chapter Participatory Action Research and Action Science Compared: A Commentary, pages 85–98. Sage, Newbury Park, CA, USA, 1991.

[AS02a] Alain Abran and Asma Sellami. Initial modeling of the measurement concepts in the iso vocabulary of terms in metrology. In *Software Measurement and Estimation — Proceedings of the 12th International Workshop on Software Measurement IWSM2002, Magdeburg, Germany*, pages 9–20, Aachen, Germany, 2002. Shaker Verlag. ISBN 3-8322-0765-1.

[AS02b] Alain Abran and Asma Sellami. Measurement and metrology requirements for empirical studies in software engineering. In *Proceedings of the 10th International Workshop on Software Technology and Engineering Practice (STEP'02)*. IEEE Computer Society, 2002.

[Aus96] Robert D. Austin, editor. *Measuring and Managing Performance in Organizations*. Dorset House Publishing, New York, NY, USA, 1996.

[BA03] Luigi Buglione and Alain Abran. Assessment of measurement indicators in software process improvement frameworks. In *Investigations in Software Measurement: Proceedings of the 13th International Workshop on*

Software Measurement (IWSM 2003), Montréal, QC, Canada, pages 287–309, Aachen, Germany, September 23–25 2003. Shaker Verlag. ISBN: 3832218807.

[BA04a] Kent Beck and Cynthia Andres. *Extreme Programming Explained: Embrace Change*. The XP Series. Addison-Wesley Professional, Boston, MA, USA, 2nd edition, November 2004. ISBN: 0321278658.

[BA04b] Luigi Buglione and Alain Abran. The software measurement body of knowledge. In *Proceedings of SMEF 2004, Software Measurement European Forum, January 28-30*, pages 84–94, Rome, Italy, 2004. ISBN 88-86674-33-3.

[BA06] Luigi Buglione and Alain Abran. Introducing root-cause analysis and orthogonal defect classification at lower cmmi maturity levels. In Alain Abran, Reiner R. Dumke, and Mercedes Ruiz, editors, *Proceedings of the International Conference on Software Process and Product Measurement (MENSURA 2006)*, pages 29–41, Cádiz, Spain, 6-8 November 2006. Servicio de Publicaciones de la Universidad de Cádiz.

[Bac94] James Bach. The immaturity of the cmm. *The American Programmer*, 7(9):13–18, September 1994.

[Bac95] James Bach. Enough about process: What we need are heroes. *IEEE Software*, 12(2):96–98, March 1995.

[Bak91] Mark D. Baker. Implementing an initial software metrics program. In *Proceedings of the IEEE 1991 National Aerospace and Electronics Conference (NAECON 1991)*, volume 3, pages 1289–1294. IEEE Aerospace and Electronics Society, May 1991.

[Bam97] Judy Bamberger. Essence of the capability maturity model. *IEEE Computer*, 30(6):112–114, June 1997.

[Bar05] Liz Barnett. Metrics for application development. Online, May 2005.

[Bas90] Victor R. Basili. Recent advances in software measurement (abstract and references for talk). In *ICSE '90: Proceedings of the 12th international conference on Software engineering*, pages 44–49, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[Bas93] Victor R. Basili. The experimental paradigm in software engineering. In Hans-Dieter Rombach, Victor R. Basili, and Richard Selby, editors, *Experimental Software Engineering Issues: Critical Assessment and Future Directions*, Heidelberg, Germany, September 1993. Springer Verlag. published as *Lecture Notes in Computer Science #*706.

[Bas96] Victor R. Basili. The role of experimentation in software engineering: past, current, and future. In *ICSE '96: Proceedings of the 18th international conference on Software engineering*, pages 442–449, Washington, DC, USA, 1996. IEEE Computer Society.

[Bau72] Frederick L. Bauer. *Software Engineers: Information Processing*. Elsevier North Holland, 1972.

[Bax98] Peter Baxter. Framework for assessing measurement process effectiveness. *INCOSE Insigth*, 1:21–23, 1998.

[BB99] Shirley A. Becker and Mitchell L. Bostelman. Aligning strategic and project measurement systems. *IEEE Software*, 16(3):46–51, May/June 1999.

[BBB+97] Vic Barnett, Roger Bate, William Bradford, Kerinia Cusick, Lesley Foster, Tom Ferguson, Rod Freudenberg, Suzanne Garcia, William Gibbs, Debbie Guagliardo, Wayne Lobbestael, Pete Malpass, Mac H. McIntyre, Richard Turner, and Richard VandeMark. An integrated product development capability maturity model, draft. Technical Report CMU/SEI-97-MM-001, SEI at CMU, Pittsburgh, PA, USA, April 1997. Only available at: ftp://ftp.sei.cmu.edu/pub/IPD.

[BBFG90] Albert L. Baker, James M. Bieman, Norman Fenton, and David A. Gustafson. A philosphy for software measurement. *Journal Of Systems and Software*, 12(3):277–281, 1990.

[BBGM87] Albert L. Baker, James M. Bieman, David A. Gustafson, and Austin C. Melton. Modeling and measuring the software development process. In *Proceedings of the 20th Hawaii International Conference on System Sciences (HICSS-20)*, volume II, pages 23–30, Western Periodicals, North Hollywood, CA, USA, January 1987.

[BC91] Pierre Bourque and Vianney Coté. An experiment in software sizing with structured analysis metrics. *Journal of Systems and Software*, 15(2):159–172, 1991.

[BC95] Victor R. Basili and Gianluigi Caldiera. Improve software quality by reusing knowledge and experience. *Sloan Management Review*, 37(1):55–64, Fall 1995. MIT Press.

[BCH+95] Barry W. Boehm, C. Clark, E. Horowitz, C. Westland, R. Madachy, and R. Selby. Cost models for future life cycle processes: Cocomo 2.0. *Annals of Software Engineering*, 1(1):57–94, 1995.

[BCR94a] Victor Basili, G. Caldiera, and H. D. Rombach. *Encyclopedia of Software Engineering*, volume 2, chapter The Goal Question Metric Approach, pages 528–532. John Wiley & Sons, Inc., New York, USA, 1994.

[BCR94b] Victor R. Basili, G. Caldiera, and H. D. Rombach. *Encyclopedia of Software Engineering*, volume 1, chapter Measurement, pages 646–661. John Wiley & Sons, 1994.

[BD93] Robert C. Bamford and William J. Deibler. Standards-comparing, contrasting iso 9001 and the sei capability maturity model. *IEEE Computer*, 26(10):68–70, Oct. 1993.

[BDR96] Lionel C. Briand, Christiane M. Differding, and H. Dieter Rombach. Practical guidelines for measurement-based process improvement. Technical Report ISERN-96-05, Fraunhofer Institute for Experimental Software Engineering, Kaiserslautern, Germany, May 1996.

[BDT96] Alfred Bröckers, Christiane Differding, and Günter Threin. The role of software process modeling in planning industrial measurement programs. In *Proceedings of the Third International Software Metrics Symposium (METRICS '96)*, pages 31–40. IEEE Computer Society, March 1996.

[BEM95a] Lionel Briand, Khaled El Emam, and Walcélio L. Melo. Ainsi: an inductive method for software process improvement: concrete steps and guidelines. Technical Report CS-TR-3498, University of Maryland at College Park, College Park, MD, USA, 1995.

[BEM95b] Lionel Briand, Khaled El Emam, and Sandro Morasca. On the application of measurement theory in software engineering. Technical Report ISERN-95-04, International Software Engineering Research Network, 1995.

[BFK+06] René Braungarten, Ayaz Farooq, Martin Kunz, Andreas Schmietendorf, and Reiner R. Dumke. Applying service-oriented software measurement to derive quality indicators of open source components. *UPGRADE - The European Journal for the Informatics Professional*, 7(1), February 2006.

[BG94] Victor Basili and Scott Green. Software process evolution at the sel. *IEEE Software*, 11(4):58–66, 1994.

[BH83] Victor R. Basili and David H. Hutchens. An empirical study of a syntactic complexity family. *IEEE Transactions on Software Engineering*, SE-9(6):664–672, November 1983.

[BH03a] Emanuel R. Baker and Peter Hantos. Implementing a successful metrics program: Using the gqm-rx concept to navigate the metrics minefield. In *Proceedings of the 15th Annual Software Technology Conference (STC) 2003*, Salt Lake City, UT, USA, April/May 2003.

[BH03b] Sarah Beecham and Tracy Hall. Building a requirements process improvement model. Technical Report 378, Department of Computer Science, University of Hertfordshire, Hertfordshire, UK, 2003.

[BHM+00] Barry W. Boehm, Ellis Horowitz, Ray Madachy, Donald Reifer, Bradford K. Clark, Bert Steece, A. Winsor Brown, Sunita Chulani, and Chris Abts. *Software Cost Estimation with Cocomo II (with CD-ROM)*. Prentice Hall Ptr, Upper Saddle River, NJ, USA, 1st edition, January 2000.

[BHR05] Sarah Beecham, Tracy Hall, and Austen Rainer. Defining a requirements process improvement model. *Software Quality Control*, 13(3):247–279, 2005.

[BJ95] Judith G. Brodman and Donna L. Johnson. Return on investment (roi) from software process improvement as measured by us industry. *Software Process: Improvement and Practice*, 1(1):35–47, 1995.

[BJ00] Michael Berry and Ross Jeffery. An instrument for assessing software measurement programs. *Empirical Software Engineering*, 5(3):183–200, November 2000.

[BJ06] Patrik Berander and Per Jönsson. A goal question metric based approach for efficient measurement framework definition. In *ISESE '06: Proceedings of the 2006 ACM/IEEE International symposium on empirical software engineering*, pages 316–325, New York, NY, USA, 2006. ACM Press.

[BJC+96] Elizabeth Bailey, Cheryl Jones, David Card, Beth Layman, Joseph Dean, and John McGarry. *Practical Software Measurement, Version 2.1*. U. S. Naval Undersea Warfare Center, Newport, RI, USA, 1996.

[BKD05] René Braungarten, Martin Kunz, and Reiner R. Dumke. An approach to classify software measurement storage facilities. Technical Report 2, University of Magdeburg, Department of Computer Science, Magdeburg, Germany, January 2005.

[BKD06] René Braungarten, Martin Kunz, and Reiner R. Dumke. Historical evolution of courses of study in computer science: A german experience report. *UPGRADE - The European Journal for the Informatics Professional*, 7(4), August 2006.

[BKF+05] René Braungarten, Martin Kunz, Aayaz Farooq, Cornelius Wille, Andreas Schmietendorf, and Reiner R. Dumke. A metrics data base maturity model. In *Proceedings of the 9th IEEE International Multitopic Conference (INMIC'2005)*, Karachi, Pakistan, December 2005. IEEE Computer Society.

[BKFD05] René Braungarten, Martin Kunz, Ayaz Farooq, and Reiner R. Dumke. Towards meaningful metrics data bases. In Alain Abran and Reiner R. Dumke, editors, *Proceedings of the 15th International Workshop on Software Measurement (IWSM2005)*, pages 1–34, Montréal, QC, Canada, 12–14 September 2005. Shaker Publishing, Aachen.

[BKW+95] Roger Bate, Dorothy Kuhn, Curt Wells, James Armitage, Gloria Clark, Kerinia Cusick, Suzanne Garcia, Mark Hanna, Robert Jones, Peter Malpass, Ilene Minnich, Hal Pierson, Tim Powell, and Al Reichner. A systems engineering capability maturity model, version 1.1. Technical Report SECMM-95-01, CMU/SEI-95-MM-003, SEI at CMU, Pittsburgh, PA, USA, November 1995.

[Bla63] Peter M. Blau. *The Dynamics of Bureaucracy: A Study of Interpersonal Relations in Two Government Agencies*. The University of Chicago Press, Chicago, IL, USA, second edition, 1963.

[BM91] Terry B. Bollinger and Clement McGowan. A critical look at software capability evaluations. *IEEE Software*, 8(4):25–41, July 1991.

[BM92] John H. Baumert and Mark S. McWhinney. Software measures and the capability maturity model. Technical Report CMU/SEI-92-TR-25, ESD-TR-92-25, SEI at CMU, Pittsburgh, PA, USA, September 1992.

[BMA96] Pierre Bourque, Marcela Maya, and Alain Abran. A sizing measure for adaptive maintenance work products. In *International Function Points Users Group — 1996 Spring Conference*, Atlanta, GA, USA, 1996.

[BMB96] Lionel C. Briand, Sandro Morasca, and Victor R. Basili. Property-based software engineering measurement. *IEEE Transactions on Software Engineering*, 22(1):68–86, January 1996.

[BMB02] Lionel C. Briand, Sandro Morasca, and Victor R. Basili. An operational process for goal-driven definition of measures. *IEEE Transactions on Software Engineerin*, 28(12):1106–1125, December 2002.

[BN95] Richard Bache and Martin Neil. *Software quality assurance and measurement: a world-wide perspective*, chapter 4 — Introducing metrics into industry: a perspective on GQM, pages 59–68. International Thomson Computer Press, London, UK, 1995.

[BO96] Adrian Burr and Mal Owen. *Statistical Methods for Software Quality: Using Metrics to Control Process and Product Quality*. International Thomson Computer Press, Boston, MA, USA, 1996.

[Boe81] Barry W. Boehm. *Software Engineering Economics*. Prentice Hall, Englewood Cliffs, NJ, USA, 1981.

[Boe84] Barry W. Boehm. Software engineering economics. *IEEE Transactions on Software Engineering*, SE-10(1), January 1984.

[Boe06] Barry W. Boehm. A view of 20th and 21st century software engineering. In *ICSE '06: Proceeding of the 28th international conference on Software engineering*, pages 12–29, New York, NY, USA, 2006. ACM Press.

[BP95] Faye Budlong and Judi Peterson. Software metrics capability evaluation guide 2.0. Technical Report AD-A325 385/3, Software Technology Support Center, Hill AFB, UT, USA, October 1995. STSC, Ogden Air Logistics Center, Hill Air Force Base, UT, USA.

[BP96a] Faye C. Budlong and Judi A. Peterson. Software metrics capability evaluation methodology and implementation. *CrossTalk*, (01), January 1996. http://www.stsc.hill.af.mil/crosstalk/1996/01/metrics.asp.

[BP96b] Paul Byrnes and Mike Phillips. Software capability evaluation version 3.0 method description. Technical Report CMU/SEI-96-TR-002, ESC-TR-96-002, SEI at CMU, Pittsburgh, PA, USA, April 1996.

[BR88] Victor R. Basili and H.-D. Rombach. The tame project: Towards improvement-oriented software environments. *IEEE Transactions on Software Engineering*, 14(6):758–773, June 1988.

[Bra05] René Braungarten. A proposal for a metrics data base maturity model. *Metrics News — Journal of GI-Interest Group on Software Metrics*, 10(1):13–27, August 2005.

[Bri31] Percy William Bridgman, editor. *Dimensional Analysis*. Yale University Press, Yale, CT, USA, revised edition edition, 1931.

[Bro75] Fred Brooks. *The Mythical Man Month*. Addison-Wesley, New York, NY, USA, 1975.

[Bro96] Norm Brown. Industrial-strength management strategies. *IEEE Software*, 13(4):94–103, 1996.

[BSC96] Ilene Burnstein, Taratip Suwanassart, and Robert Carlson. Developing a testing maturity model for software test process evaluation and improvement. In *Proceedings of the International Test Conference (ITC'96)*, pages 581–590, Los Alamitos, CA, USA, 1996. Illinois Institute of Technology, IEEE Computer Society.

[BSH86] Victor R. Basili, Richard W. Selby, and David H. Hutchens. Experimentation in software engineering. *IEEE Transactions on Software Engineering*, SE-12(7):733–743, July 1986.

[Buc90a] Fletcher J. Buckley. Establishing a standard metrics program. *IEEE Computer*, pages 85–86, June 1990.

[Buc90b] Fletcher J. Buckley. Rapid prototyping a metric program. In *Proceedings of the First International Conference on Applications of Software Measurement*, November 1990.

[Bun67] Mario Bunge, editor. *Scientific Research I and II*. Springer-Verlag, Berlin Heidelberg, Germany, 1967.

[BvSJ98] Andreas Birk, Rini van Solingen, and Janne Järvinen. 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)*, pages 93–96, Bathesda, ML, USA, November 1998.

[BW84] Victor R. Basili and D.M. Weiss. A methodology for collecting valid software engineering data. *IEEE Transactions on Software Engineering*, 10(6):728–738, June 1984.

[BWD+04] Pierre Bourque, Sibylle Wolff, Robert Dupuis, Asma Sellami, and Alain Abran. Lack of consensus on measurement in software engineering: Investigation of related issues. In Alain Abran, Manfred Bundschuh, Günter Büren, and Reiner Dumke, editors, *Prooceedings of the IWSM/MetriKon 2004*, pages 1–14. Shaker, November 2004.

[BWH96] Richard L. Baskerville and A. Trevor Wood-Harper. A critical perspective on action research as a method for information systems research. *Journal of Information Technology*, 11(3):235–246, September 1996. Routledge, Taylor & Francis Group.

[BZM77] Victor R. Basili, M. Zelkowitz, and F. McGarry. The software engineering laboratory. Technical Report TR-535, University of Maryland, College Park, May 1977.

[CA88] David Card and Bill Agresti. Measuring software design complexity. *Journal of Systems and Software*, 8(3):185–197, June 1988.

[Cam20] Norman Robert Campbell, editor. *Physics: the elements*. London, UK. Cambridge University Press, 1920. (Reprinted as Foundations of Science. New York, NY, USA: Dove, 1957).

[Car91] David Card. Understanding process improvement. *IEEE Software*, 08(4):102–103, July/August 1991.

[Car92] David Card. Capability evaluations rated highly variable. *IEEE Software*, 09(5):105–111, September/October 1992.

[Car93] David Card. What makes for effective measurement? *IEEE Software*, 10(6):94–95, 1993.

[CC93] Peter Comer and Jonathan Chard. A measurement maturity model. *Software Quality Journal*, 2(4):277–289, December 1993.

[CDS86] Samuel Daniel Conte, H. E. Dunsmore, and V. Y. Shen, editors. *Software Engineering Metrics and Models*. The Benjamin/Cummings Publishing Company, Inc., Reading, MA, USA, 1986.

[CF02] Jack Cooper and Mattew Fisher. Software acquisition capability maturity model (sa–cmm), version 1.03. Technical Report CMU/SEI-2002-TR-010, ESC-TR-2002-010, SEI at CMU, Pittsburgh, PA, USA, March 2002.

[CG90] David N. Card and Robert L. Glass. *Measuring Software Design Quality*. Prentice-Hall Inc., Englewood Cliffs, NJ, USA, 1990.

[CHM95] Bill Curtis, William Hefley, and Sally Miller. Overview of the people capability maturity model (p-cmm). Technical Report CMU/SEI-95-MM-01, SEI at CMU, Pittsburgh, PA, USA, September 1995.

[CK94] Shyam R. Chidamber and Chris F. Kemerer. A metrics suite for object oriented design. *IEEE Transactions on Software Engineering*, 20(6):476–493, June 1994.

[CKO92] Bill Curtis, Marc I. Kellner, and Jim Over. Process modeling. *Communications of the ACM*, 35(9):75–90, September 1992.

[CKS03] Mary Beth Chrissis, Mike Konrad, and Sandy Shrum. *CMMI: Guidelines for Process Integration and Product Improvement*. Addison-Wesley Professional, Boston, MA, USA, 1st edition, February 2003. ISBN: 0321154967.

[CL95] Rita J. Costello and Dar-Biau Liu. Metrics for requirements engineering. *Journal of Systems and Software*, 29(1):39–63, 1995.

[Cla02] Betsy Clark. Eight secrets of software measurement. *IEEE Software*, 19(5):12–14, 2002.

[CMWH96] François Coallier, Richard McKenzie, John F. Wilson, and Joe Hatz. Trillium — model for telecom product development & support process capability. Technical report, Bell Canada, 1996.

[Coa94a] François Coallier. How iso 9001 fits into the software world. *IEEE Software*, 11(1):98–100, January 1994.

[Coa94b] François Coallier, editor. *Trillium — Model for Telecom Product Development & Support Process Capability, Release 3.0*. Bell Canada, 1994.

[Coa95] François Coallier. Trillium: A model for the assessment of telecom product development and support capability. *IEEE Software Process Newsletter*, 3(2):3–8, Winter 1995.

[Col98] Antonio Coletta. *SPICE: The theory and practice of software process improvement and capability determination*, chapter 5 — Process assessment using SPICE: The assessment activities, pages 99–122. IEEE Computer Society, Los Alamitos, CA, USA, 1998.

[Col02] Dawn Coley. *IT Measurement: Practical Advice from the Experts*, chapter 28 — Considerations for getting maximum benefit from an enterprise-wide metrics repository, pages 455–462. Addison-Wesley Information Technology Series. Addison-Wesley, Boston, MA, USA, 2002.

[CPG+92] Anita D. Carleton, Robert E. Park, Wolfhart B. Goethert, William A. Florac, Elizabeth K. Bailey, and Shari Lawrence Pfleeger. Software measurement for dod systems: Recommendations for initial core measures. Technical Report CMU/SEI-92-TR-019 ESC-TR-92-019, Software Engineering Institute, Pittsburgh, PY, USA, September 1992.

[CPG94] Anita D. Carleton, Robert E. Park, and Wolfhart B. Goethert. The sei core measures: Background information and recommendations for use and implementation. *STSC CrossTalk*, 5, May 1994.

[Cra98] Mac Craigmyle. *SPICE: The theory and practice of software process improvement and capability determination*, chapter 6 — Process assessment using SPICE: The rating framework, pages 99–122. IEEE Computer Society, Los Alamitos, CA, USA, 1998.

[Cro79] Philip B. Crosby. *Quality Is Free (Signet Shakespeare)*. McGraw-Hill Companies, New York, NY, USA, signet shakespeare; reissue edition edition, January 1979. ISBN: 0451625854.

[Cro96] Philip B. Crosby. *Quality Is Still Free: Making Quality Certain In Uncertain Times*. McGraw-Hill Companies, New York, NY, USA, 2nd revised edition, October 1996. ISBN: 0070145326.

[Cun97] J. Barton Cunningham. Case study principles for different types of cases. *Quality and Quantity*, 31(4):401–423, November 1997.

[CVS+02] Ann Cass, Christian Völcker, Philipp Sutter, Alec Dorling, and Hans Stienen. Spice in action - experiences in tailoring and extension. In *Proceedings of the 28th Euromicro Conference (EUROMICRO'02)*, pages 352–360. IEEE Computer Society, 4-6 September 2002.

[CZ90] Randolph B. Cooper and Robert W. Zmud. Information technology implementation research: A technological diffusion approach. *Journal of Management Science*, 36(2):123–139, February 1990. The Institute of Management Sciences.

[DA99] Reiner R. Dumke and Alain Abran, editors. *Software Measurement — Current Trends in Research and Practice*. Deutscher Univertsitätsverlag, Wiesbaden, Germany, 1999.

[Das92] Michael K. Daskalantonakis. A practical view of software measurement and implementation experiences within motorola. *IEEE Transactions on Software Engineering*, 18(11):998–1010, November 1992.

[Das94] Michael K. Daskalantonakis. Achieving higher sei levels. *IEEE Software*, pages 17–24, July 1994.

[DB98] Jean-Normand Drouin and Henry Barker. *SPICE: The theory and practice of software process improvement and capability determination*, chapter 2 — Introduction to SPICE, pages 19–30. IEEE Computer Society, Los Alamitos, CA, USA, 1998.

[DB99] Carol Dekkers and Mary Bradley. It is the people who count in measurement: The truth about measurement myths. *STSC CrossTalk*, pages 12–14, June 1999.

[DB01] Donna Dunaway and Michele Baker. Analysis of cmm-based appraisal for internal process improvement (cba ipi) assessment feedback. Technical Report CMU/SEI-2001-TR-021, ESC-TR-2001-021, SEI at CMU, Pittsburgh, PA, USA, November 2001.

[DBB+06a] Reiner Dumke, René Braungarten, Martina Blazey, Heike Hegewald, Daniel Reitz, and Karsten Richter. Software process measurement and control: A measurement-based point of view of software processes. Technical Report 11, Otto-von-Guericke University, Department of Computer Science, Magdeburg, Germany, 2006.

[DBB+06b] Reiner R. Dumke, René Braungarten, Martina Blazey, Heike Hegewald, Daniel Reitz, and Karsten Richter. Structuring software process metrics - a holistic semantic network based overview. In Alain Abran, Manfred Bundschuh, and Reiner R. Dumke, editors, *Proceedings of the International Workshop on Software Measurement and DASMA Software Metrik Kongress (IWSM/MetriKon 2006)*, pages 483–498, Potsdam, Germany, 2–3 November 2006. Shaker Publishing, Aachen.

[DBH05] Reiner R. Dumke, René Braungarten, and Heike Hegewald. An iso 15939-based infrastructure supporting the it software measurement. In *Praxis der Software-Messung — Tagungsband des DASMA Software Metrik Kongresses (MetriKon 2005)*, pages 87–106, Kaiserslautern, Germany, 2005. Shaker Publishing, Aachen.

[DBK+06] Reiner R. Dumke, René Braungarten, Martin Kunz, Andreas Schmietendorf, and Cornelius Wille. Strategies and appropriateness of software measurement frameworks. In Alain Abran, Reiner R. Dumke, and Mercedes Ruiz, editors, *Proceedings of the International Conference on Software Process and Product Measurement (MENSURA 2006)*, pages 150–170, Cádiz, Spain, 6-8 November 2006. Servicio de Publicaciones de la Universidad de Cádiz.

[DBY91] Michael Daskalantonakis, Victor R. Basili, and Robert Yacobellis. A method for assessing software measurement technology. *Quality Engineering*, 3(1):27–40, 1991.

[dC97] Dennis de Champeaux. *Object-oriented development process and metrics*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1997.

[DCL+99] Christophe Debou, Daniel Courtel, H. Lambert, N. Fuchs, and M. Haux. *Better Software Practice for Business Benefit: Principles and Experiences*, chapter Alcatel's Experience with Process Improvement, pages 281–301. Wiley-IEEE Computer Society Press, Los Alamitos, CA, USA, September 1999. ISBN: 978-0-7695-0049-2.

[Deb00] Christophe Debou. Process maturity: the momentum behind metrics. *Software in Focus*, 12:2–3, June 2000.

[Dek99] Carol A. Dekkers. The secrets of highly successful measurement programs. *Cutter IT Journal*, 12(4):29–35, April 1999.

[Dek05] Ton Dekkers. Basic measurement implementation: away with the crystal ball. In Ton Dekkers, editor, *Proceedings of Software Measurement European Forum (SMEF 2005)*, pages 113–122, Rome, Italy, 2005. Sogeti Nederland B.V., The Netherlands.

[DeM82a] Tom DeMarco. *Controlling Software Projects - Management, Measurement & Estimation*. Prentice Hall PTR, Englewood Cliffs, NJ, USA, June 1982.

[Dem82b] W. Edwards Deming. *Quality, Productivity, and Competitive Position*. Massachusetts Institute of Technology, Boston, MA, USA, June 1982. ISBN: 0911379002.

[Dem86] W. Edwards Deming. *Out of the Crisis*. The MIT Press, Cambridge, MA, USA, August 1986. ISBN: 0262541157.

[DeM95] Tom DeMarco. *Why Does Software Cost So Much?: And Other Puzzles of the Information Age*. Dorset House Publishing, 1995.

[Den02] Sheila P. Dennis. *IT Measurement: Practical Advice from the Experts*, chapter 18 — Avoiding Obstacles and Common Pitfalls in the Building of an Effective Metrics Program, pages 295 –304. 1. Addison-Wesley, Boston, MA, USA, 2002.

[Der00] Esther Derby. Designing useful metrics. *Software Testing & Quality Engineering Journal*, pages 50–53, May/June 2000.

[Des94] Jean-Marc Desharnais. Statistics on function point measurement programs in 20 canadian organizations. In *Software Measurement Programs in Industry and Administration — A Joint DASMA – CIM Workshop*, Koblenz, Germany, November 1994.

[DFK98] Reiner Dumke, Erik Foltin, and Reinhard Koeppe. *Softwarequalität durch Messtools. Assessment, Messung und instrumentierte ISO 9000*. Vieweg Verlagsgesellschaft, Braunschweig, Germany, April 1998. ISBN: 3528055278.

[DFS93] Christophe Debou, Norbert Fuchs, and Heinz Saria. Selling believable technology. *IEEE Software*, 10(6):22–27, November 1993.

[Die68] Jean Dieudonne, editor. *Foundations of Modern Analysis*. Academic Press, San Diego, CA, USA, October 1968.

[Dio93] Raymond Dion. Process improvement and the corporate balance sheet. *IEEE Software*, 10(4):28–35, July 1993.

[DL81] R. A. DeMillo and R. J. Lipton. *Software Metrics*, chapter Software project forecasting, pages 77–89. MIT Press, Cambridge, MA, USA, 1981.

[DL99] Tom DeMarco and Timothy Lister, editors. *Peopleware: Productive Projects and Teams*. Dorset House Publishing, New York, NY, USA, second edition, December 1999.

[dlAMO03] Maria de los Angeles Martin and Luis Olsina. Towards an ontology for software metrics and indicators as the foundation for a cataloging web system. In *Proceedings of the First Latin American Web Congress (LA-WEB'03)*, pages 103–113. Universidad Nacional de La Pampa, November 2003.

[DLW99] Soumitra Dutta, Michael Lee, and Luk Van Wassenhove. Software engineering in europe: A study of best practices. *IEEE Software*, pages 82–90, May/June 1999.

[DM97] Bo Dahlbom and Lars Mathiassen. The future of our profession. *Communications of the ACM*, 40(6):80–89, 1997.

[DM02] Carol A. Dekkers and Patricia A. McQuaid. The dangers of using software metrics to (mis)manage. *IEEE IT Professional*, pages 24–30, March 2002.

[DN03] Ray Dawson and Andrew J. Nolan. Towards a successful software metrics programme. In *Proceedings of the Eleventh Annual International Workshop on Software Technology and Engineering Practice (STEP'04)*, pages 48–51, 2003.

[Dor93] Alec Dorling. Spice: Software process improvement and capability determination. *Software Quality Journal*, 2(4):209–224, December 1993.

[Dow86] Mark Dowson. The structure of the software process. *ACM SIGSOFT Software Engineering Notes*, 11(4):6–8, August 1986.

[Dro95] Jean-Normand Drouin. The spice project: An overview. *IEEE Software Process Newsletter*, 3(2):8–9, Winter 1995.

[Dru94] Darleen A. Druyun. New air force software metrics policy 93m-017. *STSC CrossTalk*, 4, April 1994.

[DS97] Michael Diaz and Joseph Sligo. How software process improvement helped motorola. *IEEE Software*, 14(5):75–81, September/October 1997.

[dS02] Márcio Luiz Barroso da Silveira. *IT Measurement: Practical Advice from the Experts*, chapter 5 — EDS Brazil Metrics Program: Measuring for Improvement, pages 85–96. Addison-Wesley Information Technology Series. Addison-Wesley, Boston, MA, USA, 2002.

[DSZ05] Reiner R. Dumke, Andreas Schmietendorf, and Horst Zuse. Formal descriptions of software measurement and evaluation - a short overview and evaluation. Technical Report 4, Otto-von-Guericke University, Department of Computer Science, Magdeburg, Germany, 2005.

[Dum03] Reiner R. Dumke. *Software Engineering*. Vieweg Verlag, Wiesbaden, Germany, 4th, revised and enhanced edition, 2003.

[Dum05] Reiner R. Dumke. Software measurement frameworks. In *Proceedings of the 3rd World Congress for Software Quality*, volume III, Online Supplement, pages 75–84, Munich, Germany, September 2005. International Software Quality Institute (isqi), Erlangen, Germany. ISBN: 3-9809145-3-4.

[DW88] Michael S. Deutsch and Ronald R. Willis. *Software Quality Engineering, A Total Technical Management Approach*, chapter 3, pages 31–37. Prentice Hall, Englewood Cliffs, NJ, USA, 1988.

[Dyb05] Tore Dybå. An empirical investigation of the key factors for success in software process improvement. *IEEE Transactions on Software Engineering*, 31(5):410–424, May 2005.

[Ebe97] Christof Ebert. The road to maturity: Navigating between craft and science. *IEEE Software*, 14(6):77–82, November 1997.

[Ebe99] Christof Ebert. Technical controlling and software process improvement. *Journal of Systems and Software*, 46(1):25–39, April 1999.

[EC91] Board for Software Standardization ESA and Control. Esa software engineering standards, 1991. European Space Agency, Paris, France.

[EDBS05] Christof Ebert, Reiner Dumke, Manfred Bundschuh, and Andreas Schmietendorf. *Best Practices in Software Measurement — How to use metrics to improve project and process performance*. Springer, Berlin Heidelberg, 2005.

[EDM98] Khaled El Emam, Jean-Normand Drouin, and Walcelio Melo, editors. *SPICE: The theory and practice of software process improvement and capability determination*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1998.

[EFF+99] Wolfgang Emmerich, Anthony Finkelstein, Alfonso Fuggetta, Carlo Montangero, and Jean-Claude Derniame. *Software Process: Principles, Methodology, and Technology*, chapter 2 Software Process — Standards, Assessments and Improvement, pages 15–25. Lecture Notes in Computer Science 1500. Springer-Verlag, Berlin Heidelberg, Germany, February 1999. ISBN: 3540655166.

[EG99] Khaled El Emam and Dennis R. Goldenson. An empirical review of software process assessments. Technical Report ERB-1065, National Research Council Canada, Institute for Information Technology, November 1999.

[Eji91] Lem O. Ejiogu. *Software Engineering with Formal Metrics*. QED Technical Publishing Group, Boston, MA, USA, 1991.

[EL06] Fredrik Ekdahl and Stig Larsson. Experience report: Using internal cmmi appraisals to institutionalize software development performance improvement. In *32nd EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA'06)*, pages 216–223. IEEE Computer Society, August 2006.

[EM87] Michael W. Evans and John J. Marciniak. *Software Quality Assurance and Management*, chapter 7 and 8. John Wiley & Sons, New York, NY, USA, 1987.

[EMM93] Khaled El Emam, Nadir Moukheiber, and Nazim H. Madhavji. An empirical evaluation of the g/q/m method. In *CASCON '93: Proceedings of the 1993 conference of the Centre for Advanced Studies on Collaborative research*, pages 265–289. National Research Council of Canada (NRC) and IBM Centre for Advanced Studies (CAS), IBM Press, 1993.

[FBD05] Ayaz Farooq, René Braungarten, and Reiner R. Dumke. An empirical analysis of object-oriented metrics for java technologies. In *Proceedings of the 9th IEEE International Multitopic Conference (INMIC'2005)*, Karachi, Pakistan, December 2005. IEEE Computer Society.

[FBK+06] Ayaz Farooq, René Braungarten, Martin Kunz, Andreas Schmietendorf, and Reiner R. Dumke. Towards soa-based approaches for it quality assurance. In *Proceedings of the CONQUEST 2006 - Software Quality in Service-Oriented Architectures, Berlin, Germany*, pages 45–54, Heidelberg, Germany, 2006. dpunkt.verlag.

[FC99] William A. Florac and Anita D. Carleton. *Measuring the Software Process — Statistical Process Control for Software Process Improvement*. The SEI Series in Software Engineering. Addison-Wesley Professional, New York, NY, USA, July 1999.

[Fei61] Armand V. Feigenbaum. *Total Quality Control*. McGraw-Hill Book Company, New York, NY, USA, 1961.

[Fen90] Stewart Fenick. Implementing management metrics: An army program. *IEEE Software*, pages 65–72, March 1990.

[Fen91] Norman E. Fenton. *Software Metrics: A Rigorous Approach*. Kluwer Acdemic Publishers, Boston, MA, USA, April 1991.

[Fen94] Norman E. Fenton. Software measurement: A necessary scientific basis. *IEEE Transactions on Software Engineering*, 20(3):199–206, March 1994.

[FG96] Gary Ford and Norman E. Gibbs. A mature profession of software engineering. Technical Report CMU/SEI-96-TR-004, ESC-TR-96-004, SEI at CMU, Pittsburgh, PA, USA, September 1996.

[FH92] Peter H. Feiler and Watts S. Humphrey. Software process development and enactment: Concepts and definitions. Technical Report CMU/SEI-92-TR-004, SEI at CMU, Pittsburgh, PA, USA, 1992.

[FH93] Peter H. Feiler and Watts S. Humphrey. Software process development and enactment: concepts and definitions. In *Second International Conference on the Software Process 'Continuous Software Process Improvement', Berlin, Germany*, pages 28–40, Los Alamitos, CA, USA, February 1993. SEI at CMU, IEEE Computer Society.

[Fin82] Larry Finkelstein. *Handbook of Measurement Science*, volume 1, chapter Theoretical Fundamentals: Theory and Philosophy of Measurement, pages 1–30. John Wiley & Sons, Chichester, UK, 1982.

[Fin84] Larry Finkelstein. A review of the fundamental concepts of measurement. *Measurement*, 2(1):25–34, 1984.

[FLZ05] Peter Fettke, Peter Loos, and Jörg Zwicker. Business process reference models: Survey and classification. In Ekkart Kindler and Markus Nüttgens, editors, *Proceedings of the First International Workshop on Business Process Reference Models (BPRM'05)*, Nancy, France, 5-7 September 2005.

[FMG02] Peter Fraser, James Moultire, and Mike Gregory. The use of maturity models / grids as a tool in assessing product development capability. In *Proceedings of the IEEE International Engineering Management Conference, IEMC'02*, pages 244–249, Cambridge, UK, 2002. IEEE Computer Society.

[FN99] Norman E. Fenton and Martin Neil. Software metrics: successes, failures and new directions. *Journal of Systems and Software*, 47(2-3):149–157, 1999.

[FN00] Norman E. Fenton and Martin Neil. Software metrics: roadmap. In *ICSE '00: Proceedings of the Conference on The Future of Software Engineering*, pages 357–370, New York, NY, USA, 2000. ACM Press.

[FO97] Haruhiko Fukuda and Yasuo Ohashi. A guideline for reporting results of statistical analysis in japanese journal of clinical oncology. *Japanese Journal of Clinical Oncology*, 27(3):121–127, 1997.

[Fol94] Stephan Foldes, editor. *Fundamental Structures of Algebra and Discrete Mathematics*. John Wiley & Sons, New York, NY, USA, 1994.

[Fou22] Joseph Fourier, editor. *The analytical theory of heat (A. Freeman Trans.)*. Stechert, New York, NY, USA, reprint edition, 1822. (Original work published in 1822).

[FP97] Norman E. Fenton and Shari Lawrence Pfleeger. *Software Metrics — A Rigorous and Practical Approach*. International Thomson Computer Press, London, UK, 2nd edition, 1997.

[FPC97] William A. Florac, Robert E. Park, and Anita D. Carleton. Practical software measurement: Measuring for process management and improvement. Guidebook, Online, April 1997. CMU/SEI-97-HB-003.

[FPG94] Norman Fenton, Shari Lawrence Pfleeger, and Robert L. Glass. Science and substance: A challenge to software engineers. *IEEE Software*, pages 86–95, July 1994.

[FR89] Priscilla Fowler and Stanley Rifkin. Software engineering process group guide. Technical Report CMU/SEI-90-TR-024, ESD-90-TR-225, SEI at CMU, Pittsburgh, PA, USA, September 1989.

[Fuc95] Norbert Fuchs. *Software quality assurance and measurement: a worldwide perspective*, chapter 5 — Software measurement – an evolutionary approach, pages 59–68. International Thomson Computer Press, London, UK, 1995.

[FV97] Martin D. Fraser and Vijay K. Vaishnavi. A formal specifications maturity model. *Communications of the ACM*, 40(12):95–103, December 1997.

[FW86] Norman Fenton and Robin Whitty. Axiomatic approach to software metrication through program decomposition. *Computer Journal*, 29(4):330–339, 1986.

[FW95] Norman Fenton and Robin Whitty. *Software Quality Assurance and Measurement: A Worldwide Perspective*, chapter 1 — Introduction, pages 1–20. International Thomson Publishing, London, UK, 1995.

[GBC+06] Félix García, Manuel F. Bertoa, Coral Calero, Antonio Vallecillo, Francisco Ruiz, Mario Piattini, and Marcela Genero. Towards a consistent terminology for software measurement. *Information & Software Technology*, 48(8):631–644, August 2006.

[GC87] Robert B. Grady and Deborah L. Caswell. *Software Metrics: Establishing a Company-Wide Program*. Prentice Hall PTR, Englewood Cliffs, NJ, USA, 1987.

[GCWF95] John Gaffney, Robert Cruickshank, Richard Werling, and Henry Felber. *The Software Measurement Guidebook*. International Thomson Computer Press, Boston, MA, USA, September 1995. ISBN: 1850321957.

[Geo03] Elli Georgiadou. Software process and product improvement: A historical perspective. *Cybernetics and Systems Analysis*, 39(1):125–142, January 2003.

[GGM99] Dennis R. Goldenson, Anandasivam Gopal, and Tridas Mukhopadhyay. Determinants of success in software measurement programs: Initial results. In *Proceedings of the Sixth International Software Metrics Symposium (METRICS'99)*, pages 10–21, 1999.

[GHW95] Christiane Gresse, Barbara Hoisl, and Jürgen Wüst. A process model for gqm-based measurement. Technical Report STTI-95-04-E, Software-Technologie-Transfer-Initiative Kaiserslautern, Universität Kaiserslautern, Kaiserslautern, Germany, 1995.

[Gib98] Rick Gibson. Software process modeling: Theory, results and commentary. In *Proceedings of the Thirty-First Annual Hawaii International Conference on System Sciences-Volume 3*, pages 399–408, January 1998.

[Gil92] Alan Gillies. *Software Quality, Theory and Management*. Chapman & Hall Computing, London, UK, 1992.

[GJPD99] Ross Grable, Jacquelyn Jernigan, Casey Pogue, and Dale Divis. Metrics for small projects: Experiences at the sed. *IEEE Software*, 16(2):21–29, March/April 1999.

[GJR03] Dennis R. Goldenson, Joe Jarzombek, and Terry Rout. Measurement and analysis in capability maturity model integration models and software process improvement. *STSC CrossTalk*, pages 20–24, July 2003.

[GKMG02] Anandasivam Gopal, M.S. Krishnan, Tridas Mukhopadhyay, and Dennis R. Goldenson. Measurement programs in software development: Determinants of success. *IEEE Transactions on Software Engineering*, 28(9):863–875, September 2002.

[Gla94] Robert L. Glass. The software-research crisis. *IEEE Software*, 11(6):42–47, November 1994.

[Gla95] Robert L. Glass. A structure-based critique of contemporary computing research. *Journal of Systems and Software*, 28(1):3–7, 1995.

[Gla97] Robert L. Glass. Pilot studies: what, why, and how. *Journal of Systems and Software*, 36(1):85–97, 1997.

[GMK05] Anandasivam Gopal, Tridas Mukhopadhyay, and M.S. Krishnan. The impact of institutional forces on software metrics programs. *IEEE TRANSACTIONS ON SOFTWARE ENGINEERING*, 31(8):679–694, August 2005.

[GND98] Alan W. Graydon, Risto Nevalainen, and Jean-Normand Drouin. *SPICE: The theory and practice of software process improvement and capability determination*, chapter 4 — The reference model, pages 75–98. IEEE Computer Society, Los Alamitos, CA, USA, 1998.

[Goo04] Paul Goodman. *Software Metrics — Best Practices for Successful IT Management*. Philip Jan Rothstein, FBCI, Brookfield, CT, USA, 2004. ISBN 1-931332-26-6.

[Gra81] Charles Gray, editor. *Essentials of Project Management*. Petrocelli Books, Princeton, NJ, USA, 1981.

[Gra92] Robert B. Grady. *Practical Software Metrics for Project Management and Process Improvement*. Prentice Hall, Englewood Cliffs, NJ, USA, 1992.

[Gra94] Robert B. Grady. Successfully applying software metrics. *IEEE Computer*, 27(9):18–25, September 1994.

[GRB+04] Felix García, F. Ruiz, M. F. Bertoa, C. Calero, M. Genero, L. Olsina, Martín, C. Quer, N. Condori, S. Abrahao, A. Vallecillo, and M. Piattini. An ontology for software measurement. Technical Report UCLM DIAB-04-02-2, Computer Science Department, University of Castilla-La Mancha, Spain, 2004.

[Gre05] Shane Greenstein. Not a mellifluous march to maturity. *IEEE Micro*, pages 102–104, January 2005.

[Gro06] Object Management Group. Business process modeling notation (bpmn) final adopted specification 1.0, February 2006.

[GRS98] Donald J. Gantzer, Garry Roedler, and Sarah Sheard. Measurements, standards and models. *INCOSE Insight*, 1(4a):5–9, 1998.

[GRV04] Robert L. Glass, V. Ramesh, and Iris Vessey. An analysis of research in computing disciplines. *Communications of the ACM*, 47(6):89–94, June 2004.

[GS98] Eddie M. Gray and W. L. Smith. On the limitations of software process assessment and the recognition of a required re-orientation for global process improvement. *Software Quality Control*, 7(1):21–34, 1998.

[GTW93] David A. Gustafson, Joo T. Tan, and Perla Weaver. Software measure specification. In *SIGSOFT '93: Proceedings of the 1st ACM SIGSOFT symposium on Foundations of software engineering*, pages 163–168, New York, NY, USA, 1993. ACM Press.

[Gum99] Evert Gummesson. *Qualitative Methods in Management Research*. Number 2nd. Sage Publications, Thousand Oakes, CA, USA, 1999.

[GVR02] Robert L. Glass, Iris Vessey, and V. Ramesh. Research in software engineering: an analysis of the literature. *Information and Software Techology*, 44:491–506, June 2002.

[HA05] Bill C. Hardgrave and Deborah J. Armstrong. Software process improvement: it's a journey, not a destination. *Communications of the ACM*, 48(11):93–96, November 2005.

[Ham50] Richard W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 26(2):147–160, 1950.

[Han73] Per Brinch Hansen. *Operating System Principles*. Prentice-Hall Series in Automatic Computation. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1973. ISBN: 0136378439.

[Har04] Warren Harrison. A flexible method for maintaining software metrics data: a universal metrics repository. *Journal of Systems and Software*, 72(2):225–234, 2004.

[HCR+94] James Herbsleb, Anita Carleton, James Rozum, Jane Siegel, and David Zubrow. Benefits of cmm-based software process improvement: Initial results. Technical Report CMU/SEI-94-TR-013, ESC-TR-94-013, SEI at CMU, Pittsburgh, PA, USA, August 1994.

[Hei90] Dennis Heimbinger. Proscription versus prescription in process-centered environments. In *Proceedings of the 6th International Software Process Workshop*, Hokkaido, Japan, October 1990.

[Hei01] James T. Heires. What i did last summer: A software development benchmarking case study. *IEEE Software*, pages 33–39, September/October 2001.

[Hei04] Lauren Heinz. Cmmi myths and realities. *CrossTalk*, (6):8–10, June 2004.

[Het90] Bill Hetzel. The software measurement challenge. In *Proceedings of the First International Conference on Applications of Software Measurement*, November 1990.

[Het93] Bill Hetzel. *Making Software Measurement Work: Building an Effective Measurement Program*. John Wiley & Sons, New York, NY, USA, 1993.

[HF94] Tracy Hall and Norman Fenton. Implementing software metrics — the critical success factors. *Software Quality Journal*, 3(4):195–208, December 1994.

[HF97] Tracy Hall and Norman Fenton. Implementing effective software metrics programs. *IEEE Software*, 14(2):55–64, 1997.

[HG98] James D. Herbsleb and Rebecca E. Grinter. Conceptual simplicity meets organizational complexity: Case study of a corporate metrics program. In *Proceedings of the 20th International Conference on Software Engineering (ICSE'98)*, pages 271–280, 1998.

[HJPH98] Anne Mette Jonassen Hass, Jorn Johansen, and Jan Pries-Heje. Does iso 9001 increase software development maturity? In *Proceedings of the 24th Euromicro Conference*, volume 2, pages 860–866, 25-27 August 1998.

[HK81] Sallie Henry and Dennis Kafura. Software structure metrics based on information flow. *IEEE Transactions on Software Engineering*, SE-7(5):510–518, September 1981.

[HM95] Ian J. Hayes and Brendan P. Mahony. Using units of measurement in formal specifications. *Formal Aspects of Computing*, 7(3):329–347, February 1995.

[HMK+94] Volkmar Haase, Richard Messnarz, Günter Koch, Hans J"urgen Kugler, and Paul Decrinis. Bootstrap: Fine-tuning process assessment. *IEEE Software*, 11(4):25–35, June/August 1994.

[Hof00] Douglas Hoffman. The darker side of metrics. In *Pacific Northwest Software Quality Conference, Portland, Oregon*, 2000.

[Hol02] Lori Holmes. *IT Measurement: Practical Advice from the Experts*, chapter 6 — Measurement Program Implementation Approaches, pages 97–111. 1. Addison-Wesley, addison-wesley edition, May 2002. ISBN: 0-201-74158-X.

[HS87] Watts S. Humphrey and W. L. Sweet. A method for assessing the software engineering capability of contractors. Technical Report CMU/SEI-87-TR-23, SEI at CMU, Pittsburgh, PA, USA, 1987.

[HS96] Brian Henderson-Sellers. *Object-oriented metrics: measures of complexity*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1996.

[HT01] Robin B. Hunter and Richard H. Thayer, editors. *Software Process Improvement*. IEEE Computer Society, Los Alamitos, CA, USA, 2001. ISBN: 0-7695-0999-1.

[HT03] Shihong Huang and Scott Tilley. Towards a documentation maturity model. In *SIGDOC '03: Proceedings of the 21st annual international conference on Documentation*, pages 93–99, New York, NY, USA, 2003. ACM Press.

[Hum87] Watts S. Humphrey. Characterizing the software process: A maturity framework. Technical Report CMU/SEI-87-TR-11, ADA182895, SEI at CMU, Pittsburgh, PA, USA, June 1987.

[Hum88] Watts S. Humphrey. Characterizing the software process: A maturity framework. *IEEE Software*, 5(2):73–79, March/April 1988.

[Hum89] Watts S. Humphrey. *Managing the Software Process*. Addison-Wesley, Reading, MA, USA, reprinted with corrections august 1990 edition, 1989.

[Hum96] Watts S. Humphrey. *Introduction to the Personal Software Process*. Addison-Wesley Professional, Boston, MA, USA, December 1996. ISBN: 0201548097.

[Hum99] Watts S. Humphrey. *Introduction to the Team Software Process*. Addison-Wesley Professional, Boston, MA, USA, August 1999. ISBN: 020147719X.

[Hum02] Watts S. Humphrey. Three process perspectives: Organizations, teams, and people. *Annals of Software Engineering*, 14(1-4):39–72, December 2002.

[Hum05a] Watts S. Humphrey. *PSP: A Self-Improvement Process for Software Engineers*. SEI Series in Software Engineering. Addison-Wesley Professional, Boston, MA, USA, March 2005. ISBN: 0321305493.

[Hum05b] Watts S. Humphrey. *TSP — Leading a Development Team*. SEI Series in Software Engineering. Addison-Wesley Professional, Boston, MA, USA, September 2005. ISBN: 0321349628.

[HY05] Scott Y. Harmon and Simone M. Youngblood. A proposed model for simulation validation process maturity. *JDMS*, (4):179–190, October 2005. The Society for Modeling and Simulation International.

[IEE90] Computer Society IEEE. Ieee 610.12:1990 standard glossary of software engineering terminology, 1990.

[IEE92] Computer Society IEEE. Ieee 610.12:1992 standard glossary of software engineering terminology, 1992.

[IEE98] Standards Department IEEE. Ieee 1061:1998 standard for a software quality metrics methodology, revision, 1998.

[IFP04] IFPUG. *Guidelines to Software Measurement (GSM)*, volume 2. The International Function Point Users Group (IFPUG), Princeton Junction, NJ, USA, August 2004.

[IH95] Rosalind L. Ibrahim and Iraj Hirmanpour. The subject matter of process improvement:a topic and reference source for software engineering educators and trainers. Technical Report CMU/SEI-95-TR-003, ESC-TR-95-003, SEI at CMU, Pittsburgh, PY, USA, May 1995.

[IM00] Jakob Iversen and Lars Mathiassen. Lessons from implementing a software metrics program. In *Proceedings of the 33rd Hawaii International Conference on System Sciences - 2000*, volume 7. Alborg University, Denmark, IEEE Computer Society Press, 2000.

[IM03] Jakob Iversen and Lars Mathiassen. Cultivation and engineering of a software metrics program. *Information Systems Journal*, 13(1):3–19, January 2003.

[Ima86] Masaaki Imai. *Kaizen (Ky'zen): The Key to Japanese Competitive Success*. Random House Trade, New York, NY, USA, 1st edition, November 1986. ISBN: 0394551869.

[INT06] INTACS. Intacs international assessor certification scheme. Online, November 2006. http://www.intacs.info/pdf_downloads/iNTACS06-004E%20PROC%20ASSESSOR%20CERT.pdf.

[Ish85] Kaoru Ishikawa. *What is total quality control? The Japanese Way*. Prentice Hall Ptr., Englewood Cliffs, NJ, USA, 1985.

[ISO93] ISO/IEC. *International Vocabulary of Basic and General Terms in Metrology (VIM)*. International Organization for Standardization, Geneva, Switzerland, 1993.

[ISO01] ISO/IEC. *ISO/IEC 14598 Information Technology — Software Product Evaluation*. International Organization for Standardization, Geneva, Switzerland, 2001.

[ISO02] ISO/IEC. *ISO/IEC 15939 — Information Technology — Software Engineering — Software Measurement Process*. International Organization for Standardization, Geneva, Switzerland, 2002.

[ISO04] ISO/IEC. *ISO/IEC 9126-1 Information Technology – Software engineering – Product quality – Part 1: Quality model*. International Organization for Standardization, Geneva, Switzerland, 2004.

[ISO05] ISO/IEC. The iso survey — 2005. http://www.iso.org/iso/en/iso9000-14000/pdf/survey2005.pdf, December 2005. Geneva, Siwtzerland.

[ITI06] The itil homepage. http://www.itil.ork.uk/what.htm, (last visited 24 July, 2006) 2006.

[JA97] Jean-Philippe Jacquet and Alain Abran. From software metrics to software measurement methods: A process model. In *Proceedings of the 3rd International Software Engineering Standards Symposium (ISESS '97)*, pages 128–135. Research Lab. in Software Engineering Management, Department of Computer Science, UQAM, Montreal, QC, Canada, IEEE Computer Society, 1997.

[Jak98] Allan Baktoft Jakobsen. Bottom-up process improvement tricks. *IEEE Software*, 15(1):64–68, January 1998.

[Jak00] Allan Baktoft Jakobsen. Software processes: Live and let die. *IEEE Software*, 17(03):71–75, May/June 2000.

[Jar00] Janne Jarvinen. *Measurement based continuous assessment of software engineering processes*. PhD thesis, Faculty of Science, University of Oulu, Finland, Oulu, Finland, 2000.

[Jar04] Pertti Jarvinen. *On research methods*. Opinpaja Oy, Tampere, Finland, July 2004. ISBN: 951-97113-9-2.

[JB93] Ross Jeffery and Mike Berry. A framework for evaluation and prediction of metrics program success. In *Proceedings of the 1st International Software Metrics Symposium*, pages 28–39, Los Alamitos, CA, USA, 1993. IEEE Computer Society Press.

[Jef95] Ross Jeffery. Software engineering research validation. In *APSEC '95: Proceedings of the Second Asia Pacific Software Engineering Conference*, page 522, Washington, DC, USA, 1995. IEEE Computer Society.

[JG70] Joseph M. Juran and Frank M. Gryna, editors. *Quality Planning and Analysis: From Product Development Through Use*. McGraw-Hill, New York, NY, USA, 1970.

[JMF97] Michael Jones, Uffe K. Mortensen, and Jon Fairclough. The esa software engineering standards: past, present and future. In *ISESS '97: Proceedings of the 3rd International Software Engineering Standards Symposium (ISESS '97)*, pages 119–126, Washington, DC, USA, 1-6 June 1997. IEEE Computer Society.

[JMPW93] Henry J. Johansson, Patrick McHugh, A. John Pendlebury, and William A. Wheeler. *Business Process Reengineering: Breakpoint Strategies for Market Dominance*. John Wiley & Sons, New York, NY, USA, August 1993. ISBN: 0471938831.

[Jok01] Timo Jokela. *Assessment Of User-Centred Design Processes As A Basis For Improvement Action*. PhD thesis, Faculty of Science, University of Oulu, Oulu, Finland, November 2001.

[Jon96] Capers Jones. The economics of software process improvement. *IEEE Computer*, 29(1):95–97, 1996.

[Jon01] Capers Jones. Software measurement programs and industry leadership. *STSC CrossTalk*, 14(2):4–7, February 2001.

[Jon03a] Cheryl Jones. Making measurement work. *STSC CrossTalk*, pages 15–19, January 2003.

[Jon03b] Cheryl L. Jones. Implementing a successful measurement program: Tried and true practices and tools. *Cutter IT Journal*, 16(11):12–18, November 2003.

[Juc05] George Jucan. Root cause analysis for it incidents investigation. http://hosteddocs.ittoolbox.com/GJ102105.pdf, October 2005.

[Jur03] Joseph M. Juran. *Juran on Leadership for Quality: An Executive Handbook*. Free Press, Old Tappan, NJ, USA, May 2003. ISBN: 0743255771.

[Jur06] Joseph M. Juran. *Juran On Quality By Design*. Free Press, Old Tappan, NJ, USA, revised edition, 2006. ISBN: 0029166837.

[JZ97] Ross Jeffery and Benjamin Zucker. The state of practice in software metrics. Technical Report 97/1, Centre for Advanced Emprirical Software Research (CAESAR), University of New South Wales, Sydney, Australia, 1997.

[KA83] Masao Kogure and Yoji Akao. Quality function deployment and cwqc in japan. *ASQ Quality Progress*, pages 25–29, October 1983.

[KA94] Taghi M. Khoshgoftaar and Edward B. Allen. Applications of information theory to software engineering measurement. *Software Quality Journal*, 3(2):79–103, June 1994.

[Kan95] Stephen H. Kan. *Metrics and Models in Software Quality Engineering*. Addison-Wesley, Reading, MA, USA, 1995.

[Kan00] Cem Kaner. Rethinking software metrics: Evaluating measurement schemes. *Software Testing & Quality Engineering*, 2(2):50–57, March 2000.

[Kau99] Karlheinz Kautz. Making sense of measurement for small organizations. *IEEE Software*, 16(2):14–20, March/April 1999.

[KB94] Pasi Kuvaja and Adriana Bicego. Bootstrap a european assessment methodology. *Software Quality Journal*, 3(3):117–127, September 1994.

[KB99] Ron Kenett and Emanuel Baker. *Software Process Quality : Management and Control*. Computer Aided Engineering. CRC, New York, NY, USA, January 1999. ISBN: 0824717333.

[KB04] Cem Kaner and Walter P. Bond. Software engineering metrics: What do they measure and how do we know? In *Proceedings of the 10th International Software Metrics Symposium METRICS 2004*, pages 1–12, 2004.

[KBD05] Martin Kunz, René Braungarten, and Reiner R. Dumke. Measuring elearning - a classification approach for elearning systems. In Alain Abran and Reiner R. Dumke, editors, *Proceedings of the 15th Workshop on Software Measurement (IWSM'2005)*, pages 79–94, Montréal, QC, Canada, 12–14 September 2005. Shaker Publishing, Aachen.

[KBD06] Martin Kunz, René Braungarten, and Reiner R. Dumke. Bewertungsansätze und modelle für unternehmensweite software-messinitiativen. In *Arbeitskonferenz Softwarequalität und Test (ASQT) 2006*, Klagenfurt, Austria, 14–15 September 2006.

[KBS94] Stephen H. Kan, Victor R. Basili, and Lary N. Shapiro. Software quality: An overview from the perspective of total quality management. *IBM Systems Journal*, 33(1):4–19, 1994.

[KCCHS92] Annie Kuntzmann-Combelles, Peter Comer, J. Holdsworth, and Stephen Shirlaw, editors. *A Quantitative Approach to Software Management, Europe: Application of Metrics in Industry Consortium Esprit Project*. South. Bank University, London for the ami Consortium, 1992.

[KCF+96] Mike Konrad, Mary Beth Chrissis, Jack Ferguson, Suzanne Garcia, Bill Hefley, Dave Kitson, and Mark Paulk. Capability maturity modeling at the sei. *Software Process: Improvement and Practice*, 2(1):21–34, March 1996.

[Ket06] Richard Kettelerij. Designing a measurement programme for software development projects. Master's thesis, Univeristy of Amsterdam, University of Amsterdam Faculty of Science 1098 SM Amsterdam The Netherlands, August 2006.

[KH03] Heinz K. Klein and Rudy Hirschheim. Crisis in the is field? a critical reflection on the state of the discipline. *Journal of the Association for Information Systems*, 4(10), 2003.

[KHL01] Barbara A. Kitchenham, Robert T. Hughes, and Stephen G. Linkman. Modeling software measurement data. *IEEE Transactions on Software Engineering*, 9(9):788–804, September 2001.

[Kil01] Tapani Kilpi. Implementing a software metrics program at nokia. *IEEE Software*, pages 72–77, November/December 2001.

[Kin99] Atte Kinnula. *Software Process Engineering in a multi-site organization: An architectural design of a Software Process Engineering system*. PhD thesis, University of Finland, 1999.

[Kit96a] Barbara A. Kitchenham. Desmet: A method for evaluating software engineering methods and tools. Technical Report TR96-09, Department of Computer Science, University of Keele, Keele, Staffordshire, UK, August 1996.

[Kit96b] Barbara A. Kitchenham. *Software Metrics — Measurement for Software Process Improvement*. NCC Blackwell, Oxford, UK, 1996.

[Kit96c] Barbara Ann Kitchenham. Evaluating software engineering methods and tool part 1: The evaluation context and evaluation methods. *ACM SIGSOFT Software Engineering Notes*, 21(1):11–14, January 1996.

[KJ03] Margaret Kulpa and Kent A. Johnson. *Interpreting the CMMI*. Auerbach, Boca Raton, FL. USA, book & cd-rom edition, April 2003. ISBN: 0849316545.

[KK84] John Leslie King and Kenneth L. Kraemer. Evolution and organizational information systems: an assessment of nolan's stage model. *Communications of the ACM*, 27(5):466–475, May 1984.

[KKM+94] P. Koch, Pasi Kuvaja, L. Mila, A. Krzanik, S. Bicego, and G. Saukkonen. *Software Process Assessment and Improvement: The BOOTSTRAP Approach*. Blackwell Publishers, Boston, MA, USA, July 1994. ISBN: 0631196633.

[KL87] Barbara A. Kitchenham and B. Littlewood, editors. *Measurement for Software Control and Assurance*. Elsevier Applied Science, Barking, Essex, England, 1987.

[KL99] Fred N. Kerlinger and Howard B. Lee. *Foundation of Behavioral Research*. Wadsworth Publishing, Belmont, CA, USA, August 1999.

[KLBD06] Martin Kunz, Marek Leszak, René Braungarten, and Reiner R. Dumke. Design of an integrated measurement database for telecom systems development. In Alain Abran, Manfred Bundschuh, and Reiner R. Dumke, editors, *Proceedings of the International Workshop on Software Measurement and DASMA Software Metrik Kongress (IWSM/MetriKon 2006)*, pages 471–482, Potsdam, Germany, 2–3 November 2006. Shaker Publishing, Aachen.

[KLST71] David H. Krantz, R. Duncan Luce, Patrick Suppes, and Amos Tversky, editors. *Foundations of Measurement*, volume Volume 1: Additive and Polynomial Representations. Academic Press, New York, NY, USA, 1971.

[KLST89] David H. Krantz, R. Duncan Luce, Patrick Suppes, and Amos Tversky, editors. *Foundations of Measurement*, volume Volume 2: Geometrical, Threshold, and Probabilistic Representations. Academic Press, New York, NY, USA, 1989.

[KM94] Hans-Jürgen Kugler and Richard Messnarz. From the software process to software quality: Bootstrap and iso 9000. In *Software Engineering Conference, 1994. Proceedings., 1994 First Asia-Pacific*, pages 174–182, Tokyo, Japan, 7–9 December 1994. IEEE Computer Society.

[KM01] Mira Kajko-Mattsson. *Corrective Maintenance Maturity Model: Problem Management*. PhD thesis, Department of Computer and Systems Sciences (DSV), Stockholm University and Royal Institute of Technology, Sweden, Stockholm, Sweden, 2001. ISBN Nr 91-7265-311-6.

[KN92] Robert S. Kaplan and David P. Norton. The balanced scorecard — measures that drive performance. *Harvard Business Review*, pages 71–79, January-February 1992.

[KN93] Robert S. Kaplan and David P. Norton. Putting the balanced scorecard to work. *Harvard Business Review*, pages 134–147, September-October 1993.

[Kne06] Ralf Kneuper. *CMMI*. Dpunkt Verlag, Heidelberg, Germany, 2nd, revised and enhanced edition, January 2006. ISBN: 3898643735.

[KPF95] Barbara A. Kitchenham, Shari Lawrence Pfleeger, and Norman Fenton. Towards a framework for software measurement validation. *IEEE Transactions of Software Engineering*, 21(12):929–944, December 1995.

[KPP95] Barbara A. Kitchenham, Lesley Pickard, and Shari Lawrence Pfleeger. Case studies for method and tool evaluation. *IEEE Software*, 12(4):52–62, July 1995.

[KPP+02] Barbara A. Kitchenham, Shari Lawrence Pfleeger, Lesley M. Pickard, Peter W. Jones, David C. Hoaglin, Khaled El Emam, and Jarrett Rosenberg. Preliminary guidelines for empirical research in software engineering. *IEEE Transactions On Software Engineering*, 17(8):721–734, August 2002.

[KR95] Hans-Jürgen Kugler and Santiago Rementeria. Software engineering trends in europe. Technical report, European Software Institute (ESI), Bilbao, Spain, 1995.

[Kra87] Eugene Krause. *Taxicab Geometry: An Adventure in Non-Euclidean Geometry*. Dover Publications Inc., Mineola, NY, USA, February 1987. ISBN: 0486252027.

[Kra91] K. L. Kraemer, editor. *The information systems research challenge: survey research methods*, volume 3 of *Harvard Business School Research Colloquium*. Harvard Business School Press, Boulder, CO, USA, August 1991.

[Kra01] Herb Krasner. *Accumulating the Body of Evidence for The Payoff of Software Process Improvement*, chapter 12, pages 519–539. IEEE Computer Society, 2001.

[Kri88] Jürgen Kriz, editor. *Facts and Artefacts in Social Science: An Ephistemological and Methodological Analysis of Empirical Social Science*. McGraw Hill Research, New York, NY, USA, 1988.

[KS04] Seija Komi-Sirviö. *Development and Evaluation of Software Process Improvement Methods*. PhD thesis, Faculty of Science, University of Oulu, Oulu, Finland, June 2004. VTT Publications, Espoo, Finland.

[KSBD06] Martin Kunz, Andreas Schmietendorf, René Braungarten, and Reiner R. Dumke. Serviceorientierte ausrichtung von test- und messwerkzeugen. In Andreas Schmietendorf and Reiner R. Dumke, editors, *1st Workshop Bewertungsaspekte serviceorientierter Architekturen (BSOA06)*, pages 11–20, Berlin, Germany, 24 November 2006. Private publishing venture of Otto-von-Guericke-University Magdeburg.

[KSPR01] Seija Komi-Sirviö, Päivi Parviainen, and Jussi Ronkainen. Measurement automation: Methodological background and practical solutions — a multiple case study. In *Proceedings of the Seventh International Software Metrics Symposium (METRICS '01)*, Oulu, Finland, 2001. VTT Electronics.

[KT04] Christoph Kollmar and Jörg Thamm. Assessments nach der iso 15504. *Information Management & Consulting*, (4):62–69, 2004.

[Kue00] Peter Kueng. Process performance measurement system — a tool to support process-based organizations. *Total Quality Management*, 11(1):67–85, January 2000.

[Kul00] Peter Kulik. A practical approach to software metrics. *IT Professional*, 2(1):38–42, January/February 2000.

[Kuv93] Pasi Kuvaja. Bootstrap: Europe's assessment method. *IEEE Software*, 10(3):93–95, May/June 1993.

[Kuv99] Pasi Kuvaja. Bootstrap 3.0 — a spice conformant software process assessment methodology. *Software Quality Journal*, 8(1):7–19, September 1999.

[Kuz65] Simon Smith Kuznets. *Economic growth and structure; selected essays*. W. W. Norton, New York, NY, USA, 1965.

[Kyb84] Henry E . Kyburg, editor. *Theory and Measurement*. Cambridge University Press, Cambridge, UK, 1984.

[Las06] Casper Lassenius. *Software Development Control Panels — Concepts, a Toolset and Experiences*. PhD thesis, Helsinki University of Technology, April 2006. ISBN 951228197X.

[Lav00] Luigi Lavazza. Providing automated support for the gqm measurement process. *IEEE Software*, pages 56–62, May/June 2000.

[LB06] Linda M. Laird and M. Carol Brennan. *Software Measurement and Estimation: A Practical Approach*. Quantitative Software Engineering Series. Wiley-IEEE Computer Society, New York, NY, USA, June 2006.

[LBK05] Beate List, Robert M. Bruckner, and Jochen Kapaun. Holistic software process performance measurement from the stakeholders' perspective. In *Proceedings of the 16th International Workshop on Database and Expert Systems Applications (DEXA'05)*, pages 941–947. Vienna University of Technology, Austria, IEEE Computer Society, August 2005.

[LC95] Richard L. Lynch and Kelvin F. Cross. *Measure Up!: Yardsticks for Continuous Improvement*. Blackwell Publishers, Cambridge, MA, USA, 2nd edition, June 1995. ISBN: 1557867186.

[Lig02] Peter Liggesmeyer, editor. *Software Qualität — Testen, Analysieren und Verifizieren von Software*. Spektrum Akademischer Verlag, Heidelberg; Berlin, Germany, 2002.

[Lik67] Rensis Likert. *The Human Organization: Its Management and Value*. McGraw-Hill Book Company, New York, NY, USA, 1967. ISBN: 0070378517.

[Lip92] J. Liptak. Incremental implementation of the esa software metrics programme. Technical report, European Space Research and Technology Centre, Product Assurance and Safety Department, Nordwijk, The Netherlands, 1992.

[LK06] Beate List and Birgit Korherr. An evaluation of conceptual business process modelling languages. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1532–1539, New York, NY, USA, 2006. ACM Press.

[LKST90] R. Duncan Luce, David H. Krantz, Patrick Suppes, and Amos Tversky, editors. *Foundations of Measurement*, volume Volume 3: Representation, Axiomatization, and Invariance. Academic Press, New York, NY, USA, 1990.

[LM05] María Lázaro and Esperanza Marcos. Research in software engineering: Paradigms and methods. In Jaelson Castro and Ernest Teniente, editors, *17th International Conference Advanced Information Systems Engineering CAiSE 2005, Proceedings of the CAiSE'05 Workshops, Vol. 2*, volume 2, pages 517–522, Porto, Portugal, June 2005. FEUP Edições, Porto. ISBN 972-752-077-4.

[LN84] R. Duncan Luce and Louis Narens. Classification of real measurement representations by scale type. *Measurement*, 2(1):39–44, January–March 1984.

[Lon93] Jacques Lonchamp. A structured conceptual and terminological framework for software process engineering. In *ICSP*, volume Session II: Software Process Conceptual Frameworks, pages 41–53, Berlin, Germany, 1993.

[LR99] Beth Layman and Sharon Rohde. Experiences implementing a software project measurement methodology. *Software Quality Professional*, 2(1), December 1999.

[LVJ01] Marion Lepasaar, Timo Varkoi, and Hannu Jaakkola. Models and success factors of process change. In *PROFES '01: Proceedings of the Third International Conference on Product Focused Software Process Improvement*, pages 68–77, Kaiserslautern, Germany, 2001. Springer-Verlag, Berlin Heidelberg, Germany.

[Mad91] Nazim H. Madhavji. The process cycle. *IEE Software Engineering Journal*, 6(5):134–242, September 1991.

[Mar94] John J. Marciniak, editor. *Encyclopedia of software engineering*. Wiley-Interscience, New York, NY, USA, 1994.

[Mar05] Karl Marx. *Das Kapital*, volume I and II. Dietz, Berlin, Germany, 37th edition, April 2005.

[Mas43] Abraham H. Maslow. Conflict, frustration, and the theory of threat. *Journal of Abnormal and Social Psychology*, 38:81–86, 1943.

[Mas86] B. S. Massey, editor. *Measures in Science and Engineering: Their Expression, Relation and Interpretation*. Ellis Horwood, West Sussex, UK, ellis horwood series in mathematics & its applications) edition, 1986.

[May96] Jean Mayrand. Assessment of tools in the telecommunications industry: A customer perspective. In *SAST '96: Proceedings of the Proceedings of the Fourth International Symposium on Assessment of Software Tools (SAST '96)*, pages 69–70, Washington, DC, USA, 1996. Bell Canada, IEEE Computer Society.

[MB95] Steve Masters and Carol Bothwell. Cmm appraisal framework version 1.0. Technical Report CMU/SEI-95-TR-001, ESC-TR-95-001, SEI at CMU, Pittsburgh, PA, USA, February 1995.

[MB97] Sandro Morasca and Lionel Briand. Towards a theoretical framework for measuring software attributes. In *Proceedings of the 4th International Software Metrics Symposium METRICS 1997*, Albuquerque, NM, USA, 1997.

[MB00] Manoel G. Mendonça and Victor R. Basili. Validation of an approach for improving existing measurement frameworks. *IEEE Transactions on Software Engineering*, 26(6):484–499, June 2000.

[MBBD98] Manoel Gomes Mendonça, Victor R. Basili, I. S. Bhandari, and J. Dawson. An approach to improving existing measurement frameworks. *IBM Systems Journal*, 37(4):484–501, October 1998.

[MC04] Parastoo Mohagheghi and Reidar Conradi. Exploring industrial data repositories: where software development approaches meet. In *Proceedings of the 8th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'04)*, pages 61–77, Oslo, Norway, 15 June 2004.

[McA93] Donald R. McAndrews. Establishing a software measurement process. Technical Report CMU/SEI-93-TR-16 ESC-TR-93-193, SEI at CMU, Carnegie Mellon University Pittsburgh, PY, USA, July 1993.

[McC76] Thomas J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320, December 1976.

[McC00] Steve McConnell. The best influences on software engineering. *IEEE Software*, 17(1):10–17, January/February 2000.

[McF96] Robert McFeeley. Ideal: A user's guide for software process improvement. Technical Report CMU/SEI-96-HB-001, SEI at CMU, Carnegie Mellon University Pittsburgh, PY 15213, USA, February 1996.

[McG01] John McGarry. When it comes to measuring software, every project is unique. *IEEE Software*, pages 18–21, September/October 2001.

[MCJ+01] John McGarry, David Card, Cheryl Jones, Beth Layman, Elizabeth Clark, Joseph Dean, and Fred Hall. *Practical Software Measurement: Objective Information for Decision Makers*. Addison-Wesley, Reading, MA, USA, 2001.

[MD02] Frank McGarry and Bill Decker. Attaining level 5 in cmm process maturity. *IEEE Software*, pages 87–96, November 2002.

[MD03] Maricel Medina Mora and Christian Denger. Requirement metrics. an initial literature survey on measurement approaches for requirement specifications. Technical Report 096.03/E, Fraunhofer IESE, Kaiserslautern, Germany, October 2003.

[MD04] Patricia A. McQuaid and Carol A. Dekkers. Steer clear of hazards on the road to software measurement success. *Software Quality Professional*, 6(2):27–33, March 2004.

[Mel98] Werner Mellis. Software quality management in turbulent times — are there alternatives to process oriented software quality management? *Software Quality Journal*, 7(3–4):277–295, September 1998.

[Men97] Manoel Gomes Mendonça. *An Approach to Improving Existing Measurement Frameworks in Software Development Organizations*. PhD thesis, University of Maryland, 1997.

[MG98] Dorothy McKinney and Don Gantzer. Executive use of metrics: Observations and ruminations. *INCOSE Insight*, 1(4a):10–13, 1998.

[MGBB90] Austin C. Melton, David A. Gustafson, James M. Bieman, and Albert L. Baker. A mathematical perspective for software measures research. *IEE/BCS Software Engineering Journal*, 5(5):246–254, 1990.

[Mic05] Joel Michell, editor. *Measurement in Psychology: A Critical History of a Methodological Concept*. Ideas in Context (No. 53). Cambridge University Press, Cambridge, UK, 2005.

[Mil88] Everald E. Mills. Software metrics. Online, SEI Curriculum Module SEI-CM-12-1.1, December 1988.

[Min00] Arlene Minkiewicz. Software measurement? what's in it for me? In *Proceedings of the SM /ASM 2000 Conference*, 2000.

[Min02] Ilene Minnich. Cmmi appraisal methodologies: Choosing what is right for you. *CrossTalk*, (1):7–8, January 2002.

[MIO87] John D. Musa, Anthony Iannino, and Kazuhira Okumoto. *Software Reliability: Measurement, Prediction, Application*. IEEE Computer Society, 1987.

[MM04] Stephen Marshall and Geoff Mitchell. Applying spice to e-learning: an e-learning maturity model? In *CRPIT '04: Proceedings of the sixth conference on Australian computing education*, pages 185–191, Darlinghurst, Australia, Australia, 2004. Australian Computer Society, Inc.

[Moh04] Parastoo Mohagheghi. *The Impact of Software Reuse and Incremental Development on the Quality of Large Systems*. PhD thesis, Faculty of Information Technology, Mathematics and Electrical Engineering, Norwegian University of Science and Technology, Trondheim, Norway, July 2004.

[MRW77] Jim A. McCall, Paul K. Richards, and Gene F. Walters. Factors in software quality. Technical Report NTIS AD-A049-014, 015, 055, NTIS, November 1977.

[MS91] Nazim H. Madhavji and Wilhelm Schäfer. Prism — methodology and process-oriented environment. *IEEE Transactions on Software Engineering*, 17(12):1270–1283, December 1991.

[MS03] Boris Mutafelija and Harvey Stromberg. *Systematic Process Improvement Using ISO 9001: 2000 and CMMI*. Artech House Computer Library. Artech House, Boston, MA, USA, May 2003. ISBN: 1580534872.

[MSS97] Stephen G. MacDonell, Martin J. Shepperd, and Philip J. Sallis. Metrics for database systems: An empirical study. In *Proceedings of the 4th International Software Metrics Symposium (METRICS'97)*, pages 99–107, Los Alamitos, CA, USA, 1997. University of Otago, IEEE Computer Society.

[MT99] Richard Messnarz and Colin J. Tully, editors. *Better Software Practice for Business Benefit: Principles and Experiences*. Wiley-IEEE Computer Society Press, Los Alamitos, CA, USA, September 1999. ISBN: 978-0-7695-0049-2.

[Mun95] John C. Munson. Software measurement: Problems and practice. *Annals of Software Engineering*, 1(1):255–285, December 1995. Springer Netherlands.

[Mun03] John C. Munson. *Software Engineering Measurement*. Auerbach Publications, New York, NY, USA, 2003.

[Nar84] Louis Narens, editor. *Abstract Measurement Theory*. MIT Press, Cambridge, MA, USA, 1984.

[NB93] Ronald E. Nusenoff and Dennis C. Bunde. A guidebook and a spreadsheet tool for a corporate metrics program. *Journal of Systems and Software*, 23(3):245–255, December 1993.

[Nie00] Frank Niessink. *Perspectives on Improving Software Maintenance*. PhD thesis, SIKS, Dutch Graduate School for Information and Knowledge Systems, 2000.

[NK05] Anna Gunhild Nysetvold and John Krogstie. Assessing business processing modeling languages using a generic quality framework. In *Proceedings of the Tenth International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD 2005)*, 2005.

[NL93] Louis Narens and R. Duncan Luce. Further comments on the "nonrevolution" arising from axiomatic measurement theory. *Psychological Science*, 4(2):127–130, March 1993.

[Nol73] Richard L. Nolan. Managing the computer resource: a stage hypothesis. *Communications of the ACM*, 16(7):399–405, July 1973.

[NvV98] F. Niessink and H. van Vliet. Towards mature measurement programs. In *Proceedings of the Euromicro Working Conference on Software Maintenance and Reengineering*, pages 82–88. IEEE Computer Society, 1998.

[NvV01] Frank Niessink and Hans van Vliet. Measurement program success factors revisited. *Information and Software Technology*, 43(10):617–628, August 2001.

[NWZ05] Mahmood Niazi, David Wilson, and Didar Zowghi. A maturity model for the implementation of software process improvement: an empirical study. *The Journal of Systems and Software*, 74(2):155–172, January 2005.

[NWZ06] Mahmood Niazi, David Wilson, and Didar Zowghi. Critical success factors for software process improvement implementation: an empirical study. *Software Process: Improvement and Practice*, 11(2):193–211, 2006.

[OHK89] Timothy G. Olson, Watts S. Humphrey, and Dave Kitson. Conducting sei-assisted software process assessments. Technical Report CMU/SEI-89-TR-7, ESD-89-TR-7, SEI at CMU, Pittsburgh, PA, USA, February 1989.

[OIM06] Hideto Ogasawara, Takashi Ishikawa, and Tetsuro Moriya. Practical approach to development of spi activities in a large organization: Toshiba's spi history since 2000. In *ICSE '06: Proceeding of the 28th international conference on Software engineering*, pages 595–599, New York, NY, USA, 2006. ACM Press.

[OJ97] Raymond J. Offen and D. Ross Jeffery. Establishing software measurement programs. *IEEE Software*, 14(2):45–53, 1997.

[OP96] Paul Oman and Shari Lawrence Pfleeger, editors. *Applying Software Metrics*. Wiley-IEEE Computer Society Press, New York, NY, USA, first edition, 1996.

[Ost87] Leon Osterweil. Software processes are software too. In *Proceedings of the 9th International conference on Software Engineering*, pages 2–13, Monterey, CA, USA, 1987. University of Colorado Boulder, Colorado, USA, IEEE Computer Society Press. ISBN:0-89791-216-0.

[Oul95] Martyn A. Ould. *Business Processes : Modelling and Analysis for Re-Engineering and Improvement*. John Wiley & Sons, Hoboken, NJ, USA, August 1995. ISBN 0471953520.

[PA03] Edgardo Palza and Christopher Fuhrmanand Alain Abran. Establishing a generic and multidimensional measurement repository in cmmi context. In *Proceedings of the 28th Annual NASA Goddard Software Engineering Workshop (SEW'03)*, Greenbelt, Marylind, December 2003. École de Technologie Supérieure - ETS, Québec, Canada.

[Pal87] Gabriel A. Pall. *Quality Process Management*. Prentice Hall, Englewood Cliffs, NJ, USA, 1st edition, May 1987.

[Pan03] C. Ravindranath Pandian. *Software Metrics*. Auerbach Publications, New York, NY, USA, 2003.

[Pau93a] Raymond A. Paul. Metrics to improve the us army software development process. In *Proceedings of the First International Software Metrics Symposium*, Baltimore, MD, USA, May 1993.

[Pau93b] Mark C. Paulk. Comparing iso 9001 and the capability maturity model for software. *Software Quality Journal*, 2(2):245–256, December 1993.

[Pau95] Mark C. Paulk. How iso 9001 compares with the cmm. *IEEE Software*, 12(1):74–83, January 1995.

[PBH71] Johann Pfanzagl, V. Baumann, and H. Huber. *Theory of Measurement*. Physica-Verlag, Würzburg, Germany, 2nd, revised edition, 1971.

[PC94] Daniel J. Paulish and Anita D. Carleton. Case studies of software-process-improvement measurement. *IEEE Computer*, 27(9):50–57, 1994.

[PCCW93a] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. Capability maturity model for software, version 1.1. Technical Report CMU/SEI-93-TR-024, ADA263403, SEI at CMU, Pittsburgh, PY, USA, February 1993.

[PCCW93b] Mark C. Paulk, Bill Curtis, Mary Beth Chrissis, and Charles V. Weber. Capability maturity model version 1.1. *IEEE Software*, 10(4):18–27, 1993.

[Pfl93] Shari Lawrence Pfleeger. Lessons learned in building a corporate metrics program. *IEEE Software*, 10(3):67–74, May 1993.

[Pfl95a] Shari Lawrence Pfleeger. Experimental design and analysis in software engineering: Part 2: how to set up and experiment. *ACM SIGSOFT Software Engineering Notes*, 20(1):22–26, January 1995.

[Pfl95b] Shari Lawrence Pfleeger. *Software Quality Assurance and Measurement: A Worldwide perspective*, chapter 3 - Setting up a metrics pogram in industry, pages 45–58. International Thomson Computer Press, London, UK, 1995.

[Pfl97] Shari Lawrence Pfleeger. Assessing measurement. *IEEE Software*, 14(2):25–26, March/April 1997.

[Pfl01] Shari Lawrence Pfleeger. *Software Engineering: Theory and Practice*. Prentice Hall, Englewood Cliffs, NJ, USA, 2nd, hardcover edition, February 2001.

[PFO99] Troy Pearse, Tracy Freeman, and Paul Oman. Using metrics to manage the end-game of a software project. In *METRICS '99: Proceedings of the 6th International Symposium on Software Metrics*, pages 207–215, Washington, DC, USA, 1999. IEEE Computer Society.

[PGF96] Robert E. Park, Wolfhart B. Goethert, and William A. Florac. Goal-driven software measurement — a guidebook. Handbook, Online, August 1996.

[PGW94] Robert E. Park, Wolfhart B. Goethert, and J. Todd Webb. Software cost and schedule estimating: A process improvement initiative. Special Report CMU/SEI-94-SR-3, SEI at CMU, Pittsburgh, PA, USA, May 1994.

[PGW01] Mark C. Paulk, Dennis Goldenson, and David M. White. The 2001 survey of high maturity organizations. Technical Report CMU/SEI-2001-SR-013, SEI at CMU, Pittsburgh, PA, USA, 2001.

[Pid99] Michael Pidd. Just modeling through: A rough guide to modeling. *INTERFACES*, 29(2):118–132, March-April 1999.

[Pig97] Thomas M. Pigoski. *Practical Software Maintenance - Best Practices for Managing your Software Investment*. John Wiley & Sons, 1997.

[PJCK97] Shari Lawrence Pfleeger, Ross Jeffery, Bill Curtis, and Barbara Kitchenham. Status report on software measurement. *IEEE Software*, pages 33–43, March/April 1997.

[PKCS95] Kevin Pulford, Annie Kuntzmann-Combelles, and Stephen Shirlaw, editors. *A Quantitative Approach to Software Management: The ami Handbook*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1995.

[PM90] Shari Lawrence Pfleeger and Clement McGowan. Software metrics in the process maturity framework. *Journal of Systems and Software*, 12(3):255–261, July 1990.

[PM92] Daniel J. Paulish and Karl-Heinrich Möller. *Software Metrics: A Practitioner's Guide to Improved Product Development*. IEEE Computer Society Press, Los Alamitos, CA, USA, 1st edition edition, 1992.

[PM97] Lawrence H. Putnam and Ware Myers. *Industrial Strength Software: Effective Management Using Measurement*. Institute of Electrical & Electronics Engineers, Los Alamitos, CA, USA, February 1997.

[PM03] Lawrence H. Putnam and Ware Myers. *Five Core Metrics: Intelligence behind Successful Software Management*. Dorset House Publishing Co., Inc., New York, NY, USA, 2003.

[Pot93] Colin Potts. Software-engineering research revisited. *IEEE Software*, 10(5):19–28, September 1993.

[Pow01] Antony Lee Powell. *Right on Time: Measuring, Modelling and Managing Time-Constrained Software Development*. PhD thesis, Department of Computer Science, University of York, Heslington, York, UK, August 2001.

[PPS03] Tim Perkins, Ronald Peterson, and Larry Smith. Back to the basics: Measurement and metrics. *STSC CrossTalk*, pages 9–12, December 2003.

[Pre97] Roger S. Pressman. *Software Engineering — A Practitioner's Approach*. McGraw Hill, New York, NY, USA, 4th edition, 1997.

[PSS05] Markus Prechtel, Jürgen Sellentin, and Franz Schweiggert. Test and software measures for software platforms / frameworks. In *Proceedings of the 3rd World Congress for Software Quality*, volume II, Online Supplement, pages 77–88, Munich, Germany, September 2005. International Software Quality Institute (isqi), Erlangen, Germany. ISBN: 3-9809145-3-4.

[Put78] Lawrence H. Putnam. A general empirical solution to the macro software sizing and estimating problem. *IEEE Transactions on Software Engineering*, SE-4(4):345–361, July 1978.

[PV03] Sandeep Purao and Vijay Vaishnavi. Product metrics for object-oriented systems. *ACM Comput. Surv.*, 35(2):191–221, 2003.

[PWG+93] Mark Paulk, Charlie Weber, Suzanne Garcia, Mary Beth Chrissis, and Marilyn Bush. Key practices of the capability maturity model version 1.1. Technical Report CMU/SEI-93-TR-025, ESC-TR-93-178, SEI at CMU, Pittsburgh, PA, USA, February 1993.

[Raa86] Marlein Van Raalte. *Rhythm and Metre: Towards a Systematic Description of Greek Stichic Verse*. Studies in Greek & Latin Linguistics. Van Gorcum Ltd., 1986.

[Rag95] Bryce Ragland. Measure, metric, or indicator: What's the difference? *STSC CrossTalk*, 8(3):29–30, March 1995.

[RB95] Geary A. Rummler and Alan P. Brache. *Improving Performance: How to Manage the White Space in the Organization Chart*. Jossey Bass Business and Management Series. Jossey-Bass, San Fransisco, CA, USA, 2nd edition, May 1995. ISBN: 0787900907.

[RC91] Stan Rifkin and Charles Cox. Measurement in practice. Technical Report CMU/SEI-91-TR-016 ESD-TR-91-016, SEI at CMU, Pittsburgh, PA, USA, July 1991.

[RD39] F. J. Roethlisberger and W. J. Dickson. *Management and the worker: An account of a research program conducted by the Western Electric Company, Hawthorne Works, Chicago*. Harvard University Press, Cambridge, MA, USA, 1939.

[RH96a] Linda Rosenberg and Lawrence Hyatt. Developing a successful metrics program. In *Proceedings of the 8th Annual Software Technology Conference*, Utah, USA, April 1996.

[RH96b] Linda Rosenberg and Lawrence Hyatt. Developing an effectivel metrics program. In Michael Perry, editor, *Proceedings of the Product Assurance Symposium and Software Product Assurance Workshop, 19-21 March, 1996*, number ESA SP-377, pages 213–216, Noordwijk, The Netherlands, May 1996. European Space Agency.

[RH97] Linda Rosenberg and Lawrence Hyatt. Developing a successful metrics program. In *Proceedings of the International Conference on Software Engineering (ICSE 1997)*, San Fransisco, CA, USA, November 1997.

[RH04] James J. Rooney and Lee N. Vanden Heuvel. Root cause analysis for beginners. *Quality Progress*, 37(7):45–53, July 2004.

[RHB03] Austen Rainer, Tracy Hall, and Nathan Baddoo. Persuading developers to 'buy into' software process improvement: Local opinion and empirical evidence. In *ISESE '03: Proceedings of the 2003 International Symposium on Empirical Software Engineering*, pages 326–335, Washington, DC, USA, 2003. IEEE Computer Society.

[RHGH97] Ivan Rozman, Romana Vajde Horvat, József Györkös, and Marjan Herčuko. Processus — integration of sei cmm and iso quality models. *Software Quality Journal*, 6(1):37–63, March 1997.

[RHMP85] Ronald A. Radice, John T. Harding, Paul E. Munnis, and Richard W. Phillips. A programming process study. *IBM Systems Journal*, 24(2):91–101, 1985.

[Ric04] David F. Rico. *ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers*. J. Ross Publishing, Inc., Boca Raton, FL, USA, February 2004. ISBN: 193215924X.

[Rif03] Stan Rifkin. Two good reasons why new software processes are not adopted. In *Workshop Adoption-Centric Software Engineering (ACSE 2003), Proceedings of the International Conference on Software Engineering*, Portland, OR, USA, May 2003. Master Systems Inc.

[RJ94] John Roche and Mike Jackson. Software measurement methods: Recipes for success. *Information and Software Technology*, 36(3):173–189, March 1994.

[RL01] Juan F. Ramil and Meir M. Lehman. Defining and applying metrics in the context of continuing software evolution. In *METRICS '01: Proceedings of the 7th International Symposium on Software Metrics*, page 199, Washington, DC, USA, 2001. IEEE Computer Society.

[Rob79] Fred S. Roberts. *Measurement Theory with Applications to Decision Making, Utility, and the Social Sciences*. Addison-Wesley, New York, NY, USA, 1979.

[Rob97] Daniel Robey. Research commentary: Diversity in information systems research — threat, promise, and responsibility. *Information Systems Research*, 7(4):400–408, 1997.

[Roy91] Winston Royce. Current problems. In Christine Anderson and Merlin Dorfman, editors, *Aerospace Software Engineering: A Collection of Concepts*, volume V-136 of *Progress in Astronautics and Aeronautics Series*, pages 5–15, Washington, D.C., USA, 1991. American Institute of Aeronautics and Astronautics (AIAA).

[RP88] Ronald A. Radice and Richard W. Phillips. *Software engineering: an industrial approach*, volume 1. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, January 1988. ISBN: 0138232202.

[RR85] Samuel T. Redwine and William E. Riddle. Software technology maturation. In *ICSE '85: Proceedings of the 8th international conference on Software engineering*, pages 189–200, Los Alamitos, CA, USA, 1985. IEEE Computer Society Press. ISBN:0-8186-0620-7.

[RROC85] Ronald A. Radice, Norman K. Roth, Almerin C. O'Hara, and William A. Ciarfella. A programming process architecture. *IBM Systems Journal*, 24(2):79–90, 1985.

[RS98] Terry P. Rout and Peter G. Simms. *SPICE: The theory and practice of software process improvement and capability determination*, chapter 3 — Introduction to SPICE documents and architecture, pages 31– 74. IEEE Computer Society, Los Alamitos, CA, USA, 1998.

[Rub87] Howard A. Rubin. Critical success factors for measurement programs. In *Proceedings of the 1987 Spring Conference of the International Function Point Users Group (IFPUG)*, Scottsdale, AZ, USA, 1987.

[Rub90] Howard A. Rubin. Measurement — where we've been. *The Rubin Review*, 3(3), July 1990.

[Rub93] Howard A. Rubin. Debunking metric myths. *The American Programmer*, February 1993.

[Rug93] David Rugg. Using a capability evaluation to select a contractor. *IEEE Software*, 10(4):36–45, July 1993.

[Rus02] Janet Russac. *IT Measurement: Practical Advice from the Experts*, chapter 18 — Cheaper, Better, Faster: A Measurement Program That Works, pages 147–158. 1. Addison-Wesley, Boston, MA, USA, 2002.

[Sag97] Andrew P. Sage. Systematic measurements: At the interface between information and systems management, systems engineering, and operations research. *Annals of Operations Research*, 71(0):17–35, January 1997. Springer Netherlands.

[Sai03] Hossein Saiedian. Practical recommendations to minimize software capability evaluation risks. *Software Process: Improvement and Practice*, 8(3):145–156, 2003.

[SC88] S. Siegel and N. J. Castellan, editors. *Nonparametrics Statistics for the Behavioral Sciences*. McGraw-Hill, New York, NY, USA, 2 edition, 1988.

[Sch02] Norman F. Schneidewind. Body of knowledge for software quality measurement. *IEEE Computer*, pages 77–83, February 2002.

[Sch03] Jay J. Schlickman. *ISO 9001: 2000 Quality Management System Design*. rtech House Technology Management and Professional Development. Artech House, Boston, MA, USA, January 2003. ISBN: 1580535267.

[SE78] Gerald I. Susman and Roger D. Evered. An assessment of the scientific merits of action research. *Administrative Science Quarterly*, 23(4):582–603, December 1978.

[Sed97] Carl Seddio. Applying review and product metrics to the software engineering process: a case study. *Software Quality Journal*, 1(3):133–145, September 1997. ISSN: 0963-9314.

[SEI01a] SEI. Appraisal requirements for cmmi, version 1.1 (arc, v1.1). Technical Report CMU/SEI-2001-TR-034, ESC-TR-2001-034, SEI at CMU, Pittsburgh, PA, USA, December 2001.

[SEI01b] SEI. *Standard CMMI Appraisal Method for Process Improvement (SCAMPI), Version 1.1: Method Definition Document*. Number CMU/SEI-2001-HB-001. SEI at CMU, Pittsburgh, PA, USA, December 2001.

[SEI02a] SEI. Capability maturity mode integration (cmmi), continuous representation, version 1.1. Technical Report CMU/SEI-2002-TR-012 ESC-TR-2002-012, SEI at CMU, Pittsburgh, PA, USA, March 2002.

[SEI02b] SEI. Capability maturity model integration (cmmi), staged representation, version 1.1. Technical Report CMU/SEI-2002-TR-012 ESC-TR-2002-012, SEI at CMU, Pittsburgh, PA, USA, March 2002.

[SEI06] SEI. Cmmi for development, version 1.2. Technical Report CMU/SEI-2006-TR-008, ESC-TR-2006-008, SEI at CMU, Pittsburgh, PA, USA, August 2006.

[SEL97] Hans Stienen, Franz Engelmann, and Ernst Lebsanft. Bootstrap: Five years of assessment experience. In *8th International Workshop on Software Technology and Engineering Practice (STEP'97) (including CASE'97)*, pages 371–, Los Alamitos, CA, USA, 1997. IEEE Computer Society.

[SG02] Kerstin V. Siakas and Elli Georgiadou. Empirical measurement of the effects of cultural diversity on software quality management. *Software Quality Journal*, 10(2):169–180, September 2002.

[Sha90] Mary Shaw. Prospects for an engineering discipline of software. *IEEE Software*, 7(6):15–24, 1990.

[Sha01] Mary Shaw. The coming-of-age of software architecture research. In *ICSE '01: Proceedings of the 23rd International Conference on Software Engineering*, pages 657–664, Washington, DC, USA, 2001. SEI at CMU, IEEE Computer Society.

[Sha02] Mary Shaw. What makes good research in software engineering. *International Journal of Software Tools for Technology Transfer*, 4(1):1–7, 2002.

[Sha03] Mary Shaw. Writing good software engineering research papers. In *Proceedings of the 25th International Conference on Software Engineering*, pages 726–736, Pittsburgh, PA, USA, May 2003. SEI at CMU. ISBN: 0-7695-1877-X.

[She31] Walter A. Shewhart. *Economic Control of Quality of Manufactured Product*. Van Nostrand Reinhold Company, New York, NY, USA, 1931.

[She94] C. C. Shelley. Experience of implementing software measurement programmes in industry. *Software Quality Management*, pages 95–106, 1994.

[She95] Martin Shepperd. *Foundations of Software Measurement*. Prentice Hall PTR, London, UK, 1995.

[She01] Sarah A. Sheard. Evolution of the frameworks quagmire. *IEEE Software*, pages 96–98, July 2001.

[SL74] Patrick C. Suppes and R. Duncan Luce. *Encyclopaedia Britannica*, chapter Theory of Measurement, pages 739–745. Encyclopaedia Britannica, Inc., Chicago, IL, USA, 1974.

[Slo97] Malcolm Slovin. Measuring measurement maturity. *Cutter IT Metrics Strategies*, 3(4):11–13, 1997.

[SMH96] Dirk Stelzer, Werner Mellis, and Georg Herzwurm. Software process improvement via iso 9000? results of two surveys among european software houses. In *Proceedings of the Twenty-Ninth Hawaii International Conference on System Sciences*, volume 1, pages 703–712, 3-6 January 1996.

[SPSB91] Richard W. Selby, Adam A. Porter, Doug C. Schmidt, and Jim Berney. Metric-driven analysis and feedback systems for enabling empirically guided software development. In *ICSE '91: Proceedings of the 13th international conference on Software engineering*, pages 288–298, Los Alamitos, CA, USA, 1991. IEEE Computer Society Press.

[SR05] Ian Sommerville and Jane Ransom. An empirical study of industrial requirements engineering process assessment and improvement. *ACM Transaction on Software Engineering and Methodology*, 14(1):85–117, January 2005.

[Sta02] Michael Stark. Integrating theory and practice: Applying the quality improvement paradigm to product line engineering. In *ICSE 2002: International Conference on Software Engineering*, Orlando, FL, USA, May 2002. NASA Goddard Space Flight Center.

[Ste46] Stanley Smith Stevens. On the theory of scales of measurement. *Science*, 103:677–680, 1946.

[Ste75] Stanley Smith Stevens. *Psychophysics: Introduction to its Perceptual, Neural, and Social Prospects*. John Wiley & Sons, New York, NY, USA, 1975.

[Sup72] Patrick C. Suppes, editor. *Axiomatic Set Theory*. Dover Publications, New York, NY, USA, 1972.

[SW49] Claude E. Shannon and Warren Weaver. *The mathematical theory of communication*. University of Illinois Press, Urbana, IL, USA, October 1949. ISBN: 0252725484.

[Syd82] Peter H. Sydenham, editor. *Handbook of Measurement Science*, volume 1. Wiley, New York, NY, USA, 1982.

[SZ63] Patrick Suppes and Joseph L. Zinnes. *Handbook of Mathematical Psychology*, volume 1, chapter I Basic measurement theory, pages 1–76. John Wiley, New York, NY, USA, 1963.

[Tay16] Frederick Winslow Taylor. The principles of scientific management. *Bulletin of the Taylor Society*, December 1916.

[Tay02] Christine B. Tayntor. *Six Sigma Software Development*. Auerbach Publishers Inc., Boca Raton, FL, USA, July 2002. ISBN: 0849311934.

[Tho05] Oliver Thomas. Understanding the term reference model in information systems research: History, literature analysis and explanation. In Ekkart Kindler and Markus Nüttgens, editors, *Proceedings of the First International Workshop on Business Process Reference Models (BPRM'05)*, Nancy, France, 5-7 September 2005.

[THP93] Walter F. Tichy, Nico Habermann, and Lutz Prechelt. Future directions in software engineering: Summary of the 1992 dagstuhl workshop. *ACM SIGSoft Software Engineering Notes*, 18(1), January 1993.

[Tic92] TickIT. A guide to software quality management system construction and certification using en29001. Technical Report Issue 2.0, UK Department of Trade and Industry and the British Computer Society, London, UK, 1992.

[Tic98] Walter F. Tichy. Should computer scientists experiment more? *IEEE Computer*, 31(5):32–40, May 1998.

[TLPH95] Walter F. Tichy, Paul Lukowicz, Lutz Prechelt, and Ernst A. Heinz. Experimental evaluation in computer science: A quantitative study. *Journal of Systems and Software*, 28(1):9–18, January 1995.

[Tor58] Warren S. Torgerson. *Theory and Methods of Scaling*. John Wiley & Sons, London, UK, 1958.

[UE05] Medha Umarji and Henry Emurian. Acceptance issues in metrics program implementation. In *Proceedings of the 11th IEEE International Software Metrics Symposium (METRICS 2005)*, pages 20–30. University of Maryland Baltimore County, Baltimore, ML, USA, IEEE Computer Society, September 2005.

[Ull88] Jeffrey D. Ullman, editor. *Principles of Database and Knowledge-Base Systems*, volume 1. Computer Science Press, Rockville, MD, USA, 1988.

[VCW+84] J. Vosburgh, B. Curtis, R. Wolverton, B. Albert, H. Malec, S. Hoben, and Y. Liu. Productivity factors and programming environments. In *ICSE '84: Proceedings of the 7th international conference on Software engineering*, pages 143–152, Piscataway, NJ, USA, 1984. IEEE Press.

[Vin93] Walter G. Vincenti. *What Engineers Know and How They Know It — Analytical Studies from Aeronautical History*. John Hopkins Studies in the History of Technology. John Hopkins University Press, Baltimore, MD, USA, reprint edition, February 1993.

[vLvSO+98] Frank van Latum, Rini van Solingen, Markku Oivo, Barbara Hoisl, Dieter Rombach, and Günther Ruhe. Adopting gqm-based measurement in an industrial environment. *IEEE Software*, 15(1):78–86, January 1998.

[vS04] Rini van Solingen. Measuring the roi of software process improvement. *IEEE Software*, 21(3):32–38, May/June 2004.

[vSB99] Rini van Solingen and Egon Berghout. *The Goal/Question/Metric Method*. McGraw-Hill, London, UK, 1999.

[vSB01] Rini van Solingen and Egon Berghout. Integrating goal-oriented measurement in industrial software engineering: Industrial experiences with and additions to the goal/question/metric method (gqm). In *Proceedings of the Seventh International Software Metrics Symposium (METRICS '01)*, pages 246–259, 2001.

[Wal99] Alastair J. Walker. A software quality perspective on the evolution of iso 9001:1994 to iso 9001:2000. In *Proceedings of the Fourth IEEE International Symposium and Forum on Software Engineering Standards*, pages 58–66, 17-21 May 1999.

[Wan03] Yingxu Wang. The measurement theory for software engineering. In *Canadian Conference on Electrical and Computer Engineering, 2003. IEEE CCECE 2003.*, volume 2, pages 1321–1324, Montréal, QC, Canada, May 2003.

[WB01] Terence L. Woodings and Gary A. Bundell. A framework for software project metrics. In *ESCOM'01: Proceedings of the 12th European Conference on Software Control and Metrics*, 2001.

[WBD06] Cornelius Wille, René Braungarten, and Reiner R. Dumke. Addressing drawbacks of software measurement data integration. In *Proceedings of the 3rd Software Measurement European Forum (SMEF 2006)*, Rome, Italy, 10–12 May 2006. (online part).

[WDA93] Paul F. Wilson, Larry D. Dell, and Gaylord F. Anderson. *Root Cause Analysis: A Tool for Total Quality Management*. ASQ Quality Press, Milwaukee, WI, USA, September 1993. ISBN: 0873891635.

[WDK+99] Yingxu Wang, Alec Dorling, Graham King, Margaret Ross, Jeff Staples, and Ian Court. A worldwide survey on best practices toward software engineering process excellence. *ASQ Journal of Software Quality Professional*, 2(1):34–43, December 1999.

[Wei71] Gerald M. Weinberg. *The Psychology of Computer Programming*. Van Nostrand Reinhold, New York, NY, USA, 1971.

[Wei92] Gerald M. Weinberg. *Quality Software Management: Systems Thinking*, volume 1. Dorset House Publishing Company, New York, NY, USA, 1992. ISBN: 0-932633-22-6.

[Wel94] Edward F. Weller. Using metrics to manage software projects. *IEEE Computer*, 27(9):27–33, September 1994.

[Wel00] Edward F. Weller. Practical applications of statistical process control. *IEEE Software*, 17(3):48–55, May/June 2000.

[Wes02] Linda Westfall. 12 steps to useful software metrics. In *Proceedings of the International Conference on Software Quality 2002, Ottawa, ON, Canada,*, volume 12, pages 1–11. The Westfall Team, Plano, TX, USA, 2002.

[Whi97] Scott A. Whitmire. *Object-Oriented Design Measurement*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1997.

[Wie97a] Isabella Wieczorek. On the establishment of successful measurement programs in industry. *Software Process: Improvement and Practice*, 3(3):191–194, 1997.

[Wie97b] Karl E. Wiegers. Software metrics: Ten traps to avoid. *Software Development*, 5(10), October 1997.

[Wie99] Karl E. Wiegers. A software metrics primer. *Software Development*, July 1999.

[WK00] Yingxu Wang and Graham King. *Software Engineering Processes: Principles and Applications*. CRC Press LLC, Boca Raton, FL, USA, 2000.

[WKDW99] Yingxu Wang, Graham King, Alec Dorling, and Hakan Wickberg. A unified framework for the software engineering process system standards and models. In *Software Engineering Standards, 1999. Proceedings. Fourth IEEE International Symposium and Forum on*, pages 132–141, 17-21 May 1999.

[WL02] Charles Weber and Beth Layman. Measurement maturity and the cmm: How measurement practices evolve as processes mature. *Software Quality Professional*, 4(3):6–20, June 2002.

[WM93] Charlene Walrad and Eric Moss. Measurement: The key to application development quality. *IBM Systems Journal*, 32(3):445–460, 1993.

[WNHS94] Roselyn Whitney, Elise Nawrocki, Will Hayes, and Jane Siegel. Interim profile development and trial of a method to rapidly measure software engineering maturity status. Technical Report CMU/SEI-94-TR-4, ESC-TR-94-004, SEI at CMU, Pittsburgh, PA, USA, March 1994.

[WRH+00] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering: an introduction*. Kluwer Academic Publishers, Norwell, MA, USA, 2000. ISBN: 0792386825.

[XXN+06] Ruzhi Xu, Yunjiao Xue, Peiyao Nie, Yuan Zhang, and Desheng Li. Research on cmmi-based software process metrics. In *Proceedings of the First International Multi-Symposiums on Computer and Computational Sciences (IMSCCS'06)*. IEEE Computer, 2006.

[Yin02] Robert K. Yin. *Case Study Research: Design and Methods*, volume 5 of *Applied Social Research Methods*. Sage Publications Inc., Thousand Oaks, CA, USA, 3rd edition, December 2002. ISBN: 0761925538.

[YL95] Xiaobao Yu and David Alex Lamb. Metrics applicable to software design. *Annals of Software Engineering*, 1(1):23–41, December 1995.

[Zah98] Sami Zahran. *Software Process Improvement: Practical Guidelines for Business Success*. Addison-Wesley Professional, Essex, UK, 1st edition, February 1998. ISBN: 020117782X.

[Zam01] Kamal Zuhairi Zamli. Process modeling languages: A literature review. *Malaysian Journal of Computer Science*, 14(2):26–37, December 2001.

[Zav86] Pamela Zave. Let's put more emphasis on prescriptive methods. *ACM SIGSOFT Software Engineering Notes*, 11(4):98–100, August 1986.

[Zub01] Dave Zubrow. The measurement and analysis process area in cmmi. *Newsletter of the American Society for Quality*, 2001.

[Zub03] Dave Zubrow. Current trends in the adoption of the cmmi product suite. In *Proceedings of the 27th Annual International Computer Software and Applications Conference (COMPSAC'03)*, pages 126–129, Los Alamitos, CA, USA, November 2003. IEEE Computer Society.

[Zuc95] Lin Zucconi. Software process improvement paradigms for it industry: Why the bottom-up approach fits best. In *APSEC '95: Proceedings of the Second Asia Pacific Software Engineering Conference*, page 511, Washington, DC, USA, 1995. IEEE Computer Society.

[Zul01] Richard E. Zultner. The deming way to software quality. Online, http://www.softwaredioxide.com/Channels/Events/download/zultner.pdf, 2001.

[Zus91] Horst Zuse. *Software Complexity: Measures and Methods*. Walter de Gruyter & Co., Berlin, Germany, 1991.

[Zus92] Horst Zuse. Properties of software measures. *Software Quality Journal*, 1(4):225–260, December 1992.

[Zus98] Horst Zuse. *A Framework of Software Measurement*. Walter de Gruyter & Co., Berlin, Germany, 1998.

[ZW98] Marvin V. Zelkowitz and Dolores R. Wallace. Experimental models for validating technology. *IEEE Computer*, 31(5):23–31, 1998.

[ZZW95] Wayne M. Zage, Dolores M. Zage, and C. Wilburn. Avoiding metric monsters: A design metrics approach. *Annals of Software Engineering*, 1:43–55, 1995.

Appendix A

Fundamentals of measurement theory

"The essence of mathematics is not to make simple things complicated, but to make complicated things simple."

– Stan Gudder[*] –

A.1 Introduction

Especially experimental sciences live on the opportunity to quantify certain aspects of theories in order to substantiate or reject hypotheses. Several centuries ago, Galileo Galilei (*1564 – †1642) coined the slogan: "Measure what is measurable, and what is not measurable, make it measurable." Despite not being novel, that motto is still valid today and is corroborated for instance by a recent statement: "The history of science has been, in good part, the story of quantification of initially qualitative concepts." [Bun67] But measurement, or more precisely the "discipline of measurement" called *metrology* [Pal87, p. 108] [AS02a] is also fundamental in any engineering discipline, where continuous control over the applied processes, resources and created products is key to success. Tom DeMarco's [DeM82a] paraphrase "You can't control what you can't measure" gives good evidence of its importance. If the production of software shall open out an engineering branch of software engineering, control and thus sophisticated measurement has to be applied right from the beginning. That justification of measurement via an engineering analogy is popular and often utilized in related textbooks, which use it as a *raison d'être*. [Bas90] [Kit96b]

A good deal more than two decades ago, DeMillo and Lipton [DL81] argued for paying more attention to the relevance of theory of metrology when applying measurement in software engineering. But, obviously, it has been "largely ignored by both, practitioners and researchers." [Fen94] With this criticism still being up-to-date in recent times, Kaner and Bond [KB04] argue: "Our experience with graduate and undergraduate students in our Software Metrics courses, and with practitioners that we have worked with, taught, or consulted to, is that theory is profound, deep, intimidating, and not widely enough used in practice." To counteract that springe right from the beginning, this chapter is intended to highlight basic aspects of measurement theory and general metrology.

[*]John Evans Professor of Mathematics, University of Denver, Denver, CO, USA

Reference material
When studying the technical literature, one can find interpretations of measurement theory based on representational theory with different degrees of profundity relative to the specific scientific and/or engineering backgrounds of the authors. [KLST71] [PBH71] [Rob79] [Fin82] [Syd82] [Fin84] [Kri88] [KLST89] [LKST90] Also aspects of classical information theory [SW49] contribute to a better understanding. [KA94] However, it should be noted here that representational theory is not absolutely free of dispute as it for instance under harsh critique by Michell. [Mic05]

In the area of software engineering there are mainly the two research groups of luminaries around Norman Fenton in the UK and around Horst Zuse in Germany. Having tried to explain the underlying theory and to press ahead its translation into practice, they are probably the most cited authors in that direction. Hence it is not astonishing to predominantly find publications of both groups. [Fen91] [Fen94] [FW95] [FP97] [Zus91] [Zus92] [Zus98] Beyond that, few and far between also other research groups like the one around Barbara A. Kitchenham [KL87] [KPF95] [Kit96b] [KHL01], Lionel Briand [BEM95b] [BMB96] [MB97], or John C. Munson [Mun95] [Mun03] touch aspects of measurement theory in their publications.

Solely, the book of Scott A. Whitmire [Whi97, p. 142] disrupts the idyll of exhaustive description by revealing wrong or incomplete cognitions at least in the early publications of Fenton and Zuse. In 1997 he concludes that: “No discussion on measurement theory in the context of software engineering is based on complete information.” In turn, he thoroughly addresses the problem and provides a brilliant chapter on it that mainly draws upon the three volumes *Foundations of Measurement* by Krantz et al. [KLST71, KLST89, LKST90] Thus, the structure of the section is geared to the one of Whitmire but reflects knowledge compiled, equalized, and culminated out of the several sources.

A.2 Measurement — the detour for the intelligence barrier

As initially provided by Kriz [Kri88] and later adapted in the measurement-related part of the Encyclopedia of Software Engineering [Mar94], figure A.1 illustrates the motivation for measurement efforts: There is a general mental weakness or inability of human beings, also called *intelligence barrier*, to construe observations and/or empirical statements from the real-world (*empirical relational system)* towards *empirically relevant results*. More formally, those observations are called *attributes* or *properties* of *instances* or *objects* of empirical entities. In order to overcome that weakness the representational theory of measurement is applied. With its aid, the observations from the real-world are converted by means of special procedures to their representation in the world of numbers or symbols (*numerical relational system*), thereby preserving all contained information. Then, using possibly combinations of mathematical and/or statistical procedures, *numeric results* can be extracted and on their part construed by human *interpretation*. This by-pass or detour towards empirically relevant results sometimes makes construing possible at all or at least significantly easier than without.

So, the representational approach to measurement tries to find a formal, numerical way of expressing our intuitive empirical understanding of the real-world’s background stories as an essential foundation and basic requirement for later processing in the mathematical world. Beyond the concept of empirical and numerical relations, three major components are of elevated importance: empirical relational structures, numerical relational structures, and the mapping from the empirical structure into the numerical structure.

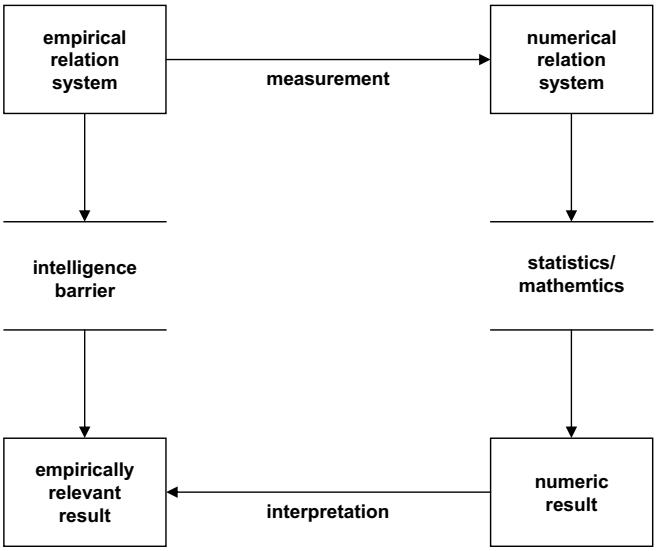

Figure A.1: Intelligence barrier (adapted from [Kri88])

A.3 Empirical and numerical relational systems

Relations
The terms *empirical relation* [FP97] or *empirical statement* [Zus98] are used to characterize the human being's understanding of things by qualitatively comparing them with already known instances of entities (e. g. 'x is taller than y') and approving them by means of direct observation. Often, there can be more than one empirical relation for the same set and while often preferred, relations need not necessarily to be binary.

In contrast to that, a *numerical relation* R on two given sets A and B (binary relation) can be defined [Whi97] as a mapping of elements of A to elements of B resulting in a set of ordered pairs (a, b) with $a \epsilon A$ and $b \epsilon B$. This is exactly the same as $(a, b) \epsilon R$, $a \mapsto b \epsilon R$ or aRb or

$$R : A \leftrightarrow B \tag{A.1}$$

Members of the source set A of a relation R are called the *domain* of R and members of the target set B of a relation R are termed *codomain* or *range*. For a complete discussion of relations and their multiple manifestations the reader could for instance refer to the exquisite book of Foldes [Fol94].

Relational systems
Relational systems or *relational structures* are member of algebraic structures, where an algebra stands for a set, called carrier set, together with a set of relations or operations and where structure refers to the conditions which are applied on the operations. [Whi97] Those structures can take the form of group structures like groups, rings and fields, vector or topological spaces as well as forms of graphs or categories. Furthermore, the term's constituent 'relational' refers to a relational kind of algebra. The operations of a relational algebra are set union, set intersection, set difference, and Cartesian product as well as concatenations, such as projection, selection, quotient, join, and semijoin. [Ull88]

Concluding, a relational system consists of a set of objects, relations between them and/or operations. As defined by Zuse [Zus91] on the basis of Roberts [Rob79], one can distinguish between empirical and numerical relational systems being important for measurement:

Let A be a non-empty carrier set of empirical objects, which are to be quantified, and R_{Aj} be ki-ary empirical relations on A with $i = 1, ..., n$, and o_{Aj} be closed binary operations with $j = 1, ..., m$ on the empirical objects in A, then, presuming a well-established empirical interpretation for all constituents of the algebra, the *empirical relational system* is defined as:

$$\mathbf{A} = (A, R_{A1}, ..., R_{An}, o_{A1}, ..., o_{Am}) \tag{A.2}$$

Typically, R_{A1} is a weak or total empirical ordering relation $\succeq$ at the same time being the most basic condition for measurement and o_{A1} is the closed binary concatenation operation $\circ$.

Let B be the non-empty set of formal objects (numbers, symbols, or structures) and R_{Br} be the ki-ary numerical relations on B with $r = 1, ..., n$ and o_{Bs} closed binary operations which on the formal objects in B with $s = 1, ..., m$ that are chosen to actually preserve the empirical relations and operations of $\mathbf{A}$ then the *numerical relational system* is defined as:

$$\mathbf{B} = (B, R_{B1}, ..., R_{Bn}, o_{B1}, ..., o_{Bm}) \tag{A.3}$$

Usually but not throughout, the domain of real numbers $\mathbb{R}$ will be the set of numbers B and R_{B1} could be no or an arbitrary like ordering relation $\geq$ on $\mathbb{R}$. In this case B is termed an ordered numerical structure. Over and above, o_{B1} could be no or an arbitrary closed binary operation $\otimes$ on $\mathbb{R}$,

A.4 Mapping between the systems

When talking about mapping, the intuitive meaning of the term 'function' will most probably come into one's mind. Universally, a function can be seen as a predefined mapping rule, which transforms an argument or input quantity into a resulting output. In the area of mathematics the term 'mapping' resides in the subarea of set theory and/or abstract algebra which is a major foundation of measurement theory. Here, the theorists distinguish mappings or rather homomorphisms for universal algebras into epimorphisms, monomorphisms, isomorphisms, endomorphisms, and automorphisms depending on the property of the mapping function. Despite being acutely important as basic knowledge, for a well understandable comprehension the reader should for instance refer to Suppes. [Sup72]

A structured approach to the mapping between the empirical and numerical relational systems requires the existence of three major components: First, *models* should be provided which are an essential aid in understanding the derivation of the mapping between the systems and in later interpreting the behavior of the formal representation. Second, there must be "basic assumptions of reality" [Zus98] characterized by so-called *axioms* that are imposed on an empirical object by the numerical representation and do not need to be proved but provide a frame of knowledge. Starting from those axioms, *theorems* [Fen91] can be logically deduced that are applied in the mapping models. Finally and third, there have to be basic *procedures* which actually allow one to assign numbers or symbols to empirical objects, that is, to construct measurements.

A.4.1 Underlying models

Without doubt, the universal but complex mapping (later called measurement) approach from the empirical to the numerical relational system requires a sequence of steps [BBFG90] [Kit96b] [Whi97] [Wan03]:

1. Identification of objects of empirical entities being of interest.
2. Identification of attributes or properties in question of those objects.
3. Development of a mapping model being appropriate for the current environment and the questioned attributes of those objects.
4. Selection of a mapping procedure qualified and built upon the mapping model.
5. Performance of the mapping itself.

As one can easily recognize in step 3, a prerequisite for the mapping between the systems is a model of the current environment and the property of interest. [Fen91] Withal, models can be generally characterized as "an abstraction of reality, allowing us to strip away detail and view an entity or concept from a particular perspective." [FP97, p. 36]

Formats: text, diagrammatic, and algorithmic models
Laird et al. [LB06] list three distinct types of models to be considered to describe the mapping between the systems: text models, diagrammatic models, and algorithmic models. Although text models can be decorated with metaphors and heuristics, among the presented ones they are least effective and powerful because describing complex relations and dynamics can be a difficult undertaking. Especially diagrammatic and algorithmic models bear the ability to clearly describe potential interrelations between entities. However, to all intents and purposes it possible to formulate more than one model, which can come in different forms like equations or diagrams, for the same questioned attribute of an object.

A.4.2 Axioms and theorems

There are three classes of axioms that can be used to frame measurement: First, there are *necessary* axioms which are mathematically important and result from the construction of the representation itself. Second, there are *non-necessary* or *structural axioms* which are used to embank the number of sets, that correctly answer the axiom set, to a manageable count. The resulting set might still require additional, complex proof of theorems. There are three sub-types of structural axioms with different requirements on the set: Those, that urge for a non-empty set, or those requiring a finite and/or countable set, or those that stipulate solvability for a class of equations. Third and finally, there are necessary *Archimedean axioms* which originate from the *Archimedean property* of the real numbers that expresses the following: For any positive number x and for any number y there is an integer n so that $n * x \geq y$ holds true. Usually, this axiom is stated as "Every strictly bounded standard sequence is finite." [Whi97] In case that requirement is not satisfied, the empirical object must not be mapped to the real numbers.

As mentioned before, theorems can be logically deduced from the axiomatic theories of measurement constraining the mapping: There are mainly three theorems important for the mapping between the systems, namely the *representation theorem*, the *uniqueness theorem*, and the meaningfulness theorem. Comprehensive and concise examinations of those theorems connected with the pertinent structures is provided by the related technical literature. [SZ63] [Rob79] [LN84]

Representation theorem
First, the representation theorem expresses the different axioms, which have to be satisfied by the empirical structure for the mapping to be meaningful, that is structure-preserving. In short, it is concerned with the problem of the "justification of the assignment of numbers to objects or phenomena." [SZ63, p. 4] Every single of the already mentioned structures of the numerical relational system has its own representation theorem, since it determines and thus articulates the conditions, under which a mapping from the empirical relational system to it is possible.

Uniqueness theorem
Second, the uniqueness theorem formulates the class of mathematical transformations which map one homomorphism to another one, thereby defining relationships or transformations between different mappings. Put another way, it deals with the tensions of "the specification of the degree to which this assignment is unique." [SZ63, p. 4] Fenton and Pfleeger [FP97] assert that this theorem affects "our ability to determine which representation is the most suitable for measuring an attribute of interest." This directly leads over to the introduction of scales and scale types which establish the uniqueness theorem.

Meaningfulness theorem
Third, the sometimes controversial meaningfulness theorem first mentioned by Suppes and Zinnes [SZ63] defines the class of statements that can be asserted in a meaningful way. Put another way, it largely deals with the invariance of statements under alternative numerical representations achieved by means of the uniqueness theorem. [KLST71] [Rob79] [LN84]

A.4.3 Measurement and measures

Generally speaking, the process of *measurement* can be defined [Ste75] [Fin84] [FW95] [FP97] [Zus98] as the mapping m from an empirical relational system $\mathbf{A}$ (cf. Equation A.2) on an numerical relational system $\mathbf{B}$ (cf. Equation A.3) by means of a formal, valid and repeatable mechanism to describe properties of objects of entities from the empirical relational system. This mechanism can be seen as "the essence of measurement: the construction of an isomorphism from some empirical structure ... onto a numerical structure." [Whi97, p. 155]

Presuming, that for the mapping m from A on B the condition $a\ R_{A1...n}\ b \Leftrightarrow m(a)\ R_{B1...u}\ m(b)$ holds true for all $a, b\ \epsilon\ A$, measurement m is defined as:

$$m : A \rightarrow B \tag{A.4}$$

Taking the above definition for granted, a *measure* is consequently the member of B assigned to a member of A by the mapping m in order to characterize an attribute while preserving the given operations and relations $R_{A1...n}$ in $R_{B1...u}$. After all, many authors [BCR94b] [FP97] unanimously define the term 'measure' textually as following: "A measure is used to characterize some property of a class of objects quantitatively."

A.4.4 Scales and scale types

Authors of the literature on representational measurement theory like e.g. Briand et al. [BEM95b] unanimously share Zuse's [Zus98] view that the ordered triple construct containing the empirical relational system $\mathbf{A} = (A, R_{A1}, ..., R_{An}, o_{A1}, ..., o_{Am})$, the numerical relational system $\mathbf{B} = (B, R_{B1}, ..., R_{Bn}, o_{B1n}, ..., o_{Bm})$, and the measurement

$m : A \rightarrow B$ is to be referred to as a *scale*

$$(\mathbf{A}, \mathbf{B}, m) \tag{A.5}$$

if and only if for all n (index of respective relations), m (index of respective operations) and for all $a_1, ..., a_k, b, c \, \epsilon \, A$ the following conditions hold true:

- $R_{An}(a_1, ..., a_k) \Leftrightarrow R_{Bn}(m(a_1), ..., m(a_k))$
- $m(a \; o_{Am} \; b) = m(a) \; o_{Bm} \; m(b)$

Taking into account the uniqueness theorem, the number of different numerical representations for a given empirical relational system shrinks with its number or empirical relations [Fen91] [FP97]. In order to know, which representations other than a found one might be acceptable, too, there is a set of admissible transformations that determines the *scale type*. [Whi97]

In his 1946 book the Harvard psychologist Stevens [Ste46] coined the terms for a set of five different scale types based and identified by the invariance of their meaning under different classes of transformations. Although, more recent literature [Alp87] [NL93] argues that these five scale types were too limited in scope and miss scale types which might exhibit periodicities, they form a *de-facto* standard in measurement theory. The scale types are sorted in ascending order with regard to uniqueness in mapping, that is, their restrictiveness: nominal, ordinal, interval, ratio, and absolute. They are respectively expressed in terms of their defining relations and/or axioms — those that remain invariant under the set of permissible transformations. [LKST90]

Nominal scale type
The usage of a nominal scale type is the most basic form of measurement and rather a kind of classification, where equivalence classes built of sequences of numbers or symbols from the carrier set B of the numerical relational system $\mathbf{B}$ are assigned to elements of the carrier set A of the empirical numerical system $\mathbf{A}$ according to a specified scheme.

While this kind of mapping does not allow to make statements about the ordering or magnitude of elements of A, the basic statement about their equivalence when having the same name after being classified is possible, anyway. [FP97] So, the only meaningful relation or operation for nominal scale types is the empirical equivalence $\approx$ relation for two objects $a, b \, \epsilon \, A$ that is reflected by the numerical equivalence relation $=$. This can be expressed formally as $a \approx b \Leftrightarrow m(a) = m(b)$. [Zus92] Putting this information into the contexts of Equation A.2 and Equation A.3, a scale of nominal type is mirrored by the ordered triple:

$$(\mathbf{A}, \mathbf{B}, m) = ((A, \approx), (B, =), m) \tag{A.6}$$

Withal, the set of allowable transformations spans on any different symbolic class denominations and can be characterized best as any one-to-one mapping that assigns the same equivalence class to equivalent instances. Because almost any transformations except combinations or confusing of class identities are permitted, only the weakest information have the ability to survive such a arbitrariness.

Ordinal scale type
A ordinal scale type distinguishes itself from the nominal scale type by the augmentation of information about the previously missing ordering of the equivalence classes with symbolic denominations. This creates the base for more sophisticated mappings.

So, in addition to statements about the empirical equivalence relation $\approx$ between two objects $a, b \, \epsilon \, A$, that scale type allows statements about the empirical ranking

and/or ordering relation $\succeq$ between a and b with respect to a special attribute which is reflected by the numerical ordering relation $\geq$. [Whi97] Because merely a ranking can be represented by ordinal scale types, so addition, subtraction, and other arithmetic operations have no meaning. Put in the formal way of Equation A.2 and Equation A.3, a scale of ordinal type is mirrored by the ordered triple:

$$(\mathbf{A}, \mathbf{B}, m) = ((A, \succeq), (B, \geq), m) \tag{A.7}$$

In order for a scale to be of ordinal type, the mapping m has to be of a weak order, that is, it has to satisfy the following axioms for all $a, b, c \, \epsilon \, A$ [Lig02]:

- $a \succeq a$ (reflexive)
- $a \succeq b$ and $b \succeq a \Rightarrow a \succeq c$ (transitive)
- $a \succeq b$ or $b \succeq a$ (connected)

The set of allowable transformations contains any monotone function (either increasing or decreasing) that preserves the empirical object's ordering. [SZ63] Thereby, a function $m(a)$ is monotone increasing if and only if for all objects a and b in the function's pertinent domain the following holds true: if $a < b,$ then $m(a) < m(b)$. Finally, a function $m(a)$ is monotone decreasing if and only if for all objects a and b in the function's pertinent domain the following holds true: if $a < b,$ then $m(a) > m(b)$.

Interval scale type
Over and above, the interval scale type enhances it predecessor in the hierarchy of Steven's scale types by retaining the interval size between two empirical objects in the course of the transformation.

Translated into formalism the following picture can be drawn on the notes of Zuse [Zus98]. Given the objects $a, b, c, d, a', b', c', d'$ which are elements of the non-empty carrier set A of the empirical relational system $\mathbf{A}$ and let $\succeq$ be a quaternary relation (e.g. the preference relation defined on intervals) on A, then the pair $(A \times A, \succeq)$ is called an *algebraic structure* if beyond weak order the following axioms are satisfied:

- if $ab \succeq cd$, then $dc \succeq ba$
- if $ab \succeq a'b'$ and $bc \succeq b'c'$, then $ac \succeq a'c'$
- if $ab \succeq cd \succeq aa$, then there exists d', d"ϵA, such that $ad' \approx cd \approx d''b$
- if $a_1, ..., a_i, ...$ is a strictly bounded sequence ($a_{i+1}a_i \approx a_2a_1$ for every a_i, a_{i+1} in the sequence; not $a_2a_1 \approx a_1a_1$; and there exist $d', d''\epsilon A$ such that $d'd'' \succeq a_ia_1 \succeq d''d'$ for all a_i in the sequence), then it is finite.

To complete the formal view of the interval scale type, the numerical relational system $\mathbf{B}$ together with its carrier set B, and the mapping m have to be placed into Equation A.2 and Equation A.3, too. While addition and subtraction are acceptable but not mandatory, multiplication and division are not. [FP97] Thus, the interval scale type can be represented formally by the ordered triple:

$$(\mathbf{A}, \mathbf{B}, m) = ((A \times A, \succeq), (B \times B, \geq), m) \tag{A.8}$$

There are two forms of allowable transformations depending on the carrier set B of the numerical relational system $\mathbf{B}$. First, there is the so-called *positive affine group* [LN84] [Whi97] defined on B as the entire set of real numbers $\mathbb{R}$ allowing transformations of the form $m(a) = ra + s$, with the coefficient $r > 0$ and the constant s. Second, when

B embraces the reduced set of positive real numbers $\mathbb{R}^+$, there is the so-called *power group* being prevalent in transformations in physics which allows transformations of the form $m(a) = ta^r$, with t as coefficient and exponent $r > 0$. In case the transformations form the power group, the scale type is renamed as *log-interval scale*. [Whi97] Withal, an admissible transformation is called *translation* if $r = 1$ or *dilation* if $r \neq 1$.

Ratio scale type
Arriving at the highest scale type of measurement, the already sophisticated mapping of the interval scale type that preserves ratio and ordering of the intervals is augmented by preserving the ratio of scale values under a transformation. This scale is common and preferred in physical sciences as well as in all-day's life. [FP97] According to Zuse [Zus98] one idea behind that type is the introduction of an additive property over a concatenation operation which increases over equal intervals. That also implies having a zero element, that is, total lack of the characteristic of interest.

Given two objects a and b being elements of the non-empty carrier set A of the empirical relational system $\mathbf{A}$, and a binary relation $\succeq$ on A, and a closed binary operation $\circ$ on A, then the ordered triple $(A, \succeq, \circ)$ is called a *closed extensive structure* if there is a function u (all arithmetics are unprohibited, now) on the domain $\mathbb{R}$ such that for all $a, b \epsilon A$ holds true [KLST71]:

- $a \succeq b \Leftrightarrow u(a) \geq u(b)$
- $u(a \circ b) = u(a) + u(b)$

Additionally, there is the condition, that a different function u' satisfies the above statements, when there is an $\alpha > 0$ such that $u'(a) = \alpha u(a)$. It is important to note, that Whitmire pointed out that there are other possibilities to achieve a ratio scale type than using closed extensive structures.

Similarly to the structure of permissible transformation for the interval scale type there exists a distinction according to domain of the carrier set B of the numerical relational system $\mathbf{B}$, as well. In case its domain completely embraces $\mathbb{R}$, transformations of the form $m(a) = a + s$ with the constant s, called *translation group* or *difference scale* [Whi97], are possible. In the other case, where $\mathbb{R}^+$ stands for B, transformations conforming to $m(a) = ra$ with the coefficient $r > 0$ are admissible. Those are then called transformations of the *similarity group*. [LN84]

Absolute scale type
The most restrictive scale type of all of those proposed by Stevens is the absolute scale type representing integer counts of elements of the carrier set A of the totally ordered empirical relational system $\mathbf{A}$. It can always be expressed in the form 'number of occurrences a in A'. Withal, the zero point is to be determined by the respective representation. [Whi97]

Expressed formally with the cognitions of Narens [Nar84], a mapping m is of an absolute scale type for $\mathbf{A}$, if there is a numerical relational system $\mathbf{B}$ that is the set of N-representations for $\mathbf{A}$ but exactly has one element:

$$(\mathbf{A}, \mathbf{B}, m) = ((A, \succeq, o_{A1}, o_{A2}, ...), (\mathbb{R}^+, \geq, \otimes_{B1}, \otimes_{B2,...}), m) \tag{A.9}$$

Although, the only admissible measurement mapping is the actual count, all arithmetics on it are meaningful. [FP97]

The set of admissible transformations can be characterized as a function of the form $m(a) = r * a$ with the constant $a = 1$. Thus, it represents a mapping on itself, also called automorphism. [Zus98]

Appropriate statistics for each scale type
A summary of the four major measurement scale types and statistics appropriate for each has been produced by Siegel and Castellan [SC88] and is reproduced in table A.1. This supports the application of measurement theory and related scale types in exhibiting, which statistical operations or tests might be applied to a measure being on hand of a special scale type.

Scale type	**Defining relations**	**Examples of appropriate statistics**	**Appropriate statistical tests**
Nominal	1. Equivalence	Mode Frequency Contingency	Non-parametric statistical tests
Ordinal	1. Equivalence 2. Greater than	Median Percentile Kendall τ Spearman r_S Kendall W	Non-parametric statistical tests
Interval	1. Equivalence 2. Greater than 3. Known ratio of any intervals	Mean Standard deviation Pearson product-moment correlation Multiple product-moment correlation	Non-parametric and parametric statistical tests
Ratio	1. Equivalence 2. Greater than 3. Known ratio of any two intervals 4. Known ratio of any two scale values	Geometric mean Coefficient of variation	Non-parametric and parametric statistical tests

Table A.1: Measurement scale types and relevant statistics (adapted from [SC88])

A.4.5 Units and dimensions

Physical sciences and engineering disciplines strongly rely on the application of units and dimensions of measurement as valuable components of measures in exposing and comparing physical systems. [HM95] Referring to the classical view of Finkelstein [Fin84] they can merely be meaningfully applied when measures use at least interval, or better ratio or absolute scale types implying to have a subset of the mathematical domain of the real numbers, $\mathbb{R}$, as the carrier set for the numerical relational system.

Melton et al. [MGBB90] render the definition of a measure (cf. equation A.4) more precisely as they regard it as a function to a set of ranked *magnitudes* with a magnitude being the product of a real number and a *unit* of measurement. Thus, units distinguish magnitudes of distinct types of quantities. [Mas86]

With a spot of contrast to the notes of Melton et al., Hayes et al. [HM95] conceive a unit as a synonym for a scale. They mention that two scales are similar, in case there is a strict ratio conversion factor between both what expels at least nominal and ordinal scale types from having meaningful units. The notion of similarity of scales that are intended to measure the same quantity is then used to define the *dimension*. Hayes et al. also are of the view that units address cognitive interests of human beings when applying arithmetic on measurement values. Units provide a system for annotations of those values with a particular symbol or sequence of symbols as error checking device against foolish combination errors. Moreover the unit symbol can serve as a guide, which operations are admissible. When algebra of equations with variables of real values shall be applied, Hayes et. al put forward to focus on the check of dimensions for *dimensional invariance* by means of *dimensional analysis* which has was first put in formal form by Fourier [Fou22] and is exhaustively discussed in the literature. [Bri31]

Again with a slightly different flavor but having the same intention of ratio conversion in mind, Wang [Wan03] comprehends a *unit* μ as "the minimum differences between two marks of a scale", that is:

$$\mu = \omega_i - \omega_{i-1} \tag{A.10}$$

Additionally, he also cherishes units of measurements since they assign a "physical or cognitive meaning of a measure on a specific attribute" of objects of the numerical relational system and/or measurement result.

After all, Kitchenham [KPF95] [Kit96b] brings up that units can be defined at least in four different ways: They can either be defined by reference to a standard and/or example, by reference to a certain theory, by reference to conversion from another unit, or by reference to a model incorporating other attributes. Moreover, she supposes practically to regard a measurement unit as an indicator of the way we measure a certain attribute of interest.

A.5 Distinguishing measurement

Fundamental and derived measurement

When talking about measurement the technical literature [Tor58] [Rob79] [Fen91] [FP97] [Zus98] distinguishes between *fundamental* and *derived* or its alternative denominations, *direct* and *indirect* or *internal* and *external*, measurement. This viewpoint again strongly adheres to Campbell [Cam20, p. 14], who initially proposed this distinction: "Measurement is fundamental if it involves no previous measurement. If it does it is derived." On that basis, Krantz et al. [KLST71] for instance are of the opinion that a measure is fundamental, if it directly characterizes an empirical attribute of the object in question and does not require the quantification of one or more other properties. In contrast, Kitchenham [Kit96b] describes indirect measures as those, that can be obtained with the aid of mathematical expressions which involve other direct measures.

In the same breath, Kitchenham cautions against a number of special problems connected with indirect measures, since the distinction of usage between a scalar or vector is non-trivial. She accompanies the problems with the illustrative example of measuring position as a vector in a Cartesian space, but distance as a scalar: When the coordinates are transposed, their distance to the origin remains untouched, while their position has obviously changed. Several years before, Conte et al. [CDS86] already pointed at those problems and used the term 'composite' measures instead of indirect ones.

Finally, Roberts [Rob79] subscribes to the view that direct measurement is being performed at early stages of scientific development, while derived measurement usually takes place later and avails itself of existing direct measures when constructing derived ones. This opinion implies that later, derived measurement should be more sophisticated than earlier, direct measurement because the underlying scientific concepts should have been improved over time. Fenton [Fen91] agrees and cites Kyburg [Kyb84] as proof, who asserts and demonstrates via the example of speed that more accurate measurements may be achieved indirectly.

Extensive and intensive measurement
Measurement theorists like Munson [Mun03] follow the notes of Campbell [Cam20], who initially distinguished between two categories of measurement, namely *intensive* or *qualitative* and *extensive* or *quantitative* measurement. With the aid of extensive measurement quantities, which represent properties for each of which there is a similar empirical operation with regard to a arithmetical additive operation. [SZ63] This enables one to answer questions like how much or how many. [Mun03] In contrast, intensive measurement deals with the quantification of intensive and/or qualitative properties that can be characterized by an absence of an additive operation. Usually, by means of qualitative measurement the degree of the relationship between instances of an entity can be determined.

Subjective and objective measurement
After all, the technical literature mentions the category of *subjective* and *objective* or *algorithmic* measurement. Conte et al. [CDS86, p. 18] state: "An objective, or algorithmic, measurement is one that can be computed precisely according to an algorithm. Its value does not change due to changes in time, place, or observer." Taking into account those eligible properties of measurement, there seems to be a myriad of disadvantages for the opposite of objective measurement, that is, subjective measurement. It is non-replicable, because it is, per se, based on the cognitions of an observer yielding most probably to different results, when performed by different individuals. Even when the same observer should be asked to replicate the measurement, it is not sure to gain the same results, the second time. Additionally, the involvement of human observers is not contemporary due to an elevated cost factor when compared with automated device. However subjective measurement owns its place and cannot be evaded e. g. for intensive measurement.

A.6 Procedures of measurement

Usually, measurement falls back on one of the following procedures as described by Whitmire [Whi97]: ordinal measurement, counting of units, or solving of inequalities. Over and above, authors like Suppes and Zinnes [SZ63] also describe pointer measurement procedures.

Zuse [Zus98] goes a more stringent way in describing types of measures reflecting different sub-procedures of measurement, when he lists: counting one attribute, additive measure, hybrid measure by additivity, hybrid measure additive plus a constant, etc. In order not to exceed the given scope of the thesis, these types shall only be referred to but not explained in detail.

Ordinal measurement
The concept of ordinal measurement tries to find representation objects in the numeric world which exactly mirror the relative order of a set of empirical objects concerning a certain attribute by an identified ordering relation. In doing so, the number sequence has to exactly match the object sequence. Zuse [Zus98] terms ordinal measurement as *ranking*.

Counting of units
Counting units depends on an empirical concatenation operation as well as on the compilation of a standard sequence of empirical objects. This assignment method is also known as extensive measurement and at the same time the foundation for measurement in the whole physical science.

Solving of inequalities
There might by circumstances, in which one might not be able to compile a standard sequence of empirical objects, but a set of inequalities describing the sequence. Then, one can try to map any empirical element on a formal element and check for potential concatenation operations like numeric addition in real number domain. If at all, any simultaneous solution of the transformed mathematical inequalities is a valid representation, then. However, two essential assumptions must hold true: The numbers assigned to the elements must be additive and the ratios of the numerical assignments must be unique, no matter what unit chosen.

Pointer measurement
Over and above, one can frequently find the point of view in the technical literature [SZ63] that so-called pointer measurement, either fundamental or derived, is a special procedure of measurement based on direct reading of some validated instrument. It starts from the attitude that measurement instruments replicate complex measurement and are validated by showing to yield values that correspond to those of some fundamental or derived mapping procedure.

A.7 Measurement issues

A.7.1 Measurement error

Similarly to laboratory experiments of i. e. chemical processes in clean-room environments that work as the theory explains but fail, when outside influences perish the ideal environment, measurement attempts are most often thwarted by noise of natural origin. The art in measurement is, to be aware of that noise and reduce it as much as possible, since the absence of noise is a sublime desire but will only occur in an ideal world.

Munson [Mun03] describes two categories of sources of measurement error to be taken into account: *methodological (systematic)* and *propagation errors*. The former one represents the introduction of bias or measurement error owing to a problem with the method of collecting the measurements. He mentions three different types, as there are: variations in the measurement instrument or tool, errors inherent in the applied method, or truncation and/or rounding errors introduced by the usage of constrained computers. Propagation errors emerge, when cumulative effects of measurement errors materialize. As a result accuracy suffers more and more from computation to computation.

Stephen H. Kan [Kan95] and Laird et al. [LB06] maintain the notion of methodological and/or systematical errors but dismiss propagation errors. Moreover, they introduce the notion of *random errors* which cannot be explained logically and occur sporadically. However, because noise or rather these disturbances are thought to be random, that is, positive errors are as likely to occur as negative errors, they will cancel each other when computing the average in the long run.

Kan provides a short mathematical summary of the topic that shall not be detained, here: Let M be the observed and/or measured score, T be the true score as it would occur in an ideal world, s be the systematic error, and ε the random error, then

$$M = T + s + \varepsilon \tag{A.11}$$

In Equation A.11 the *absolute* measurement error is the sum of terms s and ε. But more often, one is interested rather in the *relative* measurement error [Mun03]. The relative measurement error can be easily gained by shifting around the equation, so that it can be computed as the quotient of the absolute measurement error and the observed score M.

By and large, these sources of error are being described by the chapter *Theory of Measurement* in the Encyclopedia Britannica [SL74], too.

A.7.2 Validity and reliability

For measurement to be meaningful, its *validity* and *reliability* (in traditional engineering disciplines called *accuracy* and *precision* [JG70]) are important factors.

Validity
However, Munson [Mun03, p. 28] uses the term 'accuracy' of a particular measurement and describes it as "the closeness of an answer to the correct value". Other authors [Kan95] [FP97] [LB06] mention that validity refers to the fact whether mapping (model) does really express what it is thought to express. Therefore, they refer to a classification of validity into:

- *Construct validity* (correct construction of the measurement procedure),
- *Criterion-related validity* (the measurement procedure's ability to predict future behavior of the abstract concept measured), and
- *Content validity* (the extent to which a measure covers the entire range of meanings associated with the abstract concept).

Kan also states that the systematic measurement error leads to invalidity of yielded measures.

Reliability
On the other hand, at the same place Munson describes 'precision' as the "the number of significant digits in our estimate for a population value." After all, Kan connects reliability with the amount of measurement error in different measures taken and refers to an index of variation that is computed as the quotient of the measure's standard deviation and its mean.

Laird et al. [LB06] summarize the topic to the point with a practical example of a watch working without irregularity, which some people might have set five minutes ahead of time. Then, the watch provides a reliable measure of time — but with a lack of validity.

A.8 Conclusion

The chapter started with the general motivation of human beings to bypass the intelligence barrier by using the detour of a mapping from an empirical relational system to a numerical one. That procedure enables one to apply statistics and mathematics yielding to a numeric result, which can then be interpreted towards an empirically relevant result. Subsequently, it was dwelled on more theoretical aspects of the mapping between the two systems. Here, the focus was not only on the underlying models and axioms, but also on the fundamentals of measurement and measures, the possible scales and scale types as well as the importance of units and dimensions. After all, this chapter dealt with ways expressing, how to distinguish and perform measurement. Moreover, measurement issues such as measurement errors and the difference between validity and reliability of the measurement results have been pointed out.

Appendix B

A glimpse of mainstream models for SPA/SPI

"The only source of knowledge is experience."

– Albert Einstein –*

B.1 Introduction

Over the time, numerous software and system process standards, recommended practices, guidelines, capability maturity models, and the like have been proposed by different standardization bodies. As illustrated in Sheard's [She01] article 'Evolution of the Frameworks Quagmire' a continual coming and going prevails with only a few of them standing up to the ravages of time resulting in organizations being spoilt for choice. However, an elitist sphere of the probably most widely established models for SPA/SPI exists. [MT99] Previous worldwide studies by other researchers [KR95] [WDK+99] indicate that ISO 9001 spearheads the popularity, followed by the CMM of SEI, and ISO/IEC Standard 15504 (SPICE). In addition to these, some regional and/or industry sector specific models that apparently have been clearly influenced by the CMM [HT01] enjoy a good reputation: The European BOOTSTRAP and the Canadian TRILLIUM model for use in telecommunication industry. Since recently, CMM's promising successor CMMI that combines several auxiliary special-purpose maturity models of the SEI stands the test of time.

Strikingly, the Software Productivity Research (SPR) model of Capers Jones [Jon96] formidably competes with these mainstream models.

Beneath a brief analysis of major characteristics of the above listed mainstream models made up of general information as well as the respectively pursued process modeling, SPA and/or SPI approaches is presented in chronological order.

B.2 CMM v1.1

The Capability Maturity Model (CMM) was developed at the U.S. government-funded SEI at Carnegie Mellon University on the lines of Deming's notion of SPC to produce an assessment methodology for the DoD to choose among software development contractors. [Hum87, HS87] Once the method was documented properly [PCCW93a,

*German physicist who went into American exile, developer of the special and general theories of relativity, and winner of the 1921 Nobel Prize in physics, *1879 - †1955

PCCW93b] [PWG+93] [Pau93b] it very soon became also a compendium for subcontractor process improvement initiatives and later even for any organization. After the latest 1993 CMM version 1.1, the model was bit-by-bit complemented by supplementary models of the SEI and was renamed to SW-CMM. Before further development was stopped with only a draft of version 2.0 in 1998, the following supplementary models had been produced:

- The People Capability Maturity Model (P-CMM) [CHM95] was designed to help organizations in improving the ability to attract, form, motivate, structure, and keep the talents required for improving capabilities in software development.
- The Systems Engineering Capability Model (SECM) [BKW+95] was produced to foster measurement and enhancement of organization's performance to effectively translate the requirements of customers into excellent products.
- The Integrated Product Development CMM (IPD-CMM) [BBB+97] draft was prepared to improve the timely collaboration of product development disciplines.
- The Software Acquisition CMM (SA-CMM) [CF02] was created to support U.S. government software acquisition managers and improve their acquisition projects.

The SW-CMM is the oldest, most renowned model which has helped organizations such as Raytheon [Dio93], Siemens [PC94], Motorola [Das94] [DS97], Alcatel [DCL+99], Computer Sciences Corporation [MD02], or Toshiba [OIM06] all around the world in effectively and continuously improving their software processes. However, the method has its critics, too: Card [Car91] complains about the inferred, synthetic benchmark approach due to the absence of excellent organizations in the model's design stage. Bollinger et al. [BM91] are bothered by the dubiety of the factory paradigm underlying Deming's SPC approach for software production, because the assumption of replication risks does not hold. Others [Bac94] [Bam97] [Mel98] [KS04] [HA05] provide similar lists of obstacles and come to the conclusion that it should be better treated as a list of issues to look after rather than to be implemented by the letter, what most often results in inflexibility. Due to the bad performance of the IBM Corporation residing at SW-CMM's highest capability (maturity) level five in the 1990ies, the approach is generally put into question because it does obviously not guarantee excellence.

Process modeling approach
The SW-CMM was built with the premise to create a process model of the empirical, descriptive category. [KCF+96] From the process system taxonomy point of view, the model is comprised of four kinds of elements thereby leaving out the element of subsystems: SW-CMM as the process system, capability maturity levels form the process categories, KPAs constitute the single processes and key practices are equivalent to the general practices. All in all, the SW-CMM arranges 316 key practices around 18 KPAs that are organized by five distinct levels of process capability maturity. In order to model all perspectives of the process (functional, behavioral, organizational, and informational) for the organizational as well as the development and management domains, the SW-CMM's authors selected a natural language description instead of a virtual PDL.

SPA approach
In 1995 the SEI released a framework for the development, definition, and usage of appraisal methods for the SW-CMM with the denomination *CMM Appraisal Framework (CAF)*. [MB95] Because previous SW-CMM appraisals as described by Olson et al.

[OHK89] were seen as highly variable [Car92], the requirements and demanded aspects of a *CMM-based Appraisal Methods (CBAs)* were listed in that document to foster consistency and reliability of the appraisal results. Withal, it covered the appraisal families in terms of customers in need of accrediting software suppliers, suppliers in need for internal SPI, or joint SPI or risk management efforts between both groups. By laying down clearly specified rules for a compliant appraisal methodology, the SEI countered the uncontrolled growth of assessment methods predating the SW-CMM. Consequently, two forms of process appraisals have been manifested to ensure comparability:

- *CBA-IPI* [DB01]
- *SCE* [BP96b]

Both methods are highly structured, team-based, on-site document reviewing and interviewing examinations of an organization's software processes. The difference lies in the expectations: The CBA-IPI method is voluntarily, confidentially, and in most cases by representatives of the development organization itself performed to reveal and foster improvement potentials in areas being truly in need. [Rug93] This relaxed appraisal is commonly called *self-assessment*. [Coa94a] In contrast, the SCE method is a downstream *validation* of the development organization's self-assessment by SEI-trained and duly accredited (U.S. governmental) assessors, that re-check documents and examine answers during interviews. [Rug93] For the method that turns out to be a license to print money, but predominantly for the assessment body, several problems and also improvements have been considered. [Sai03] Provided that the examined areas and the observed time interval are nearly congruent, the results of both appraisal methods should be consistent. To monitor the status of organizational process maturity in between of the appraisals with one of the mentioned methods, so-called *interim profiles* [WNHS94] are meaningful.

Process maturity model
The first part of CMM's process assessment model, the process maturity model, is constructed of a practice performance scale that is based on completing a questionnaire of level-specific key questions acting as sensors with 101 process-related questions that offer decisions concerning process conformance spanning on conformance appraisal values of 'Yes', 'No', 'Doesn't apply', and 'Don't know'. [Jar00] Using a questionnaire is typically regarded as problematic, because of issues like interpretative leeway of the semantics of the questions and the inadequacy of sampling. [GS98] Furthermore, SW-CMM's process maturity scale takes the shape of five levels of software process capability maturity following the staged concept proposed by Crosby [Cro79], the SW-CMM assigns the KPAs to five distinct levels of process maturity: 1 – Initial, 2 – Repeatable, 3 – Defined, 4 – Managed, 5 – Optimizing. With the exception of the entry *capability maturity level* one, the hierarchical architecture of the SW-CMM follows the scheme, which is illustrated in figure B.1. From level two on, each maturity level comprises a set of process *goals*. Accomplishing these goals with the aid of the KPAs for each maturity level is a step towards getting processes under (statistical) control and a precondition for capability improvement. Then, each of those KPAs comprises five sections, the so-called *common features*. After all, common features outline *key practices* which collectively lead to accomplishment of the KPA's process goals.

Be it as it may, the SW-CMM stretches its performance ratings on the scopes of practice and process. Despite dismissing ratings on the project level, the organizational scope is satisfied by an assignment of a capability maturity level. [WK00]

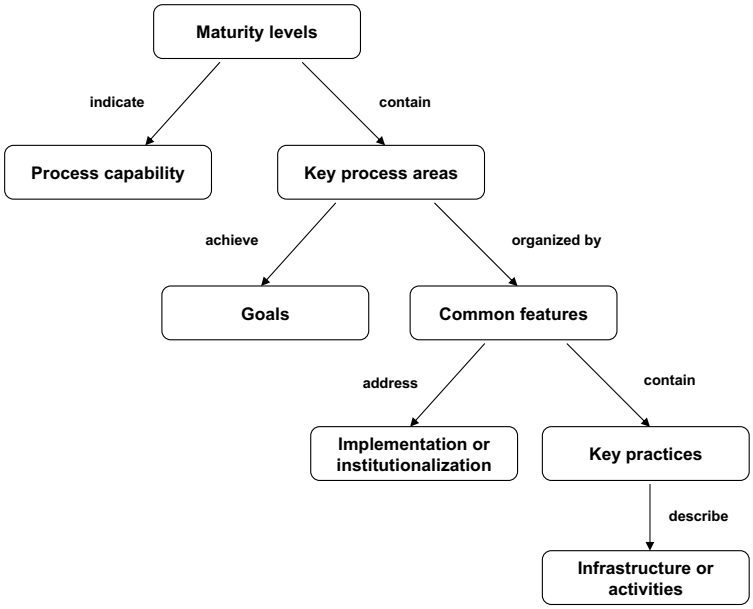

Figure B.1: The structural elements of the SW-CMM (adapted from [PCCW93a, p. 29])

Process maturity determination method
The second part of the process assessment model, the process maturity determination method, distinguishes between critical questions of the process maturity model, of which at least 90% have to be answered positively, and non-critical questions, of which merely 80% positive answers are required [BD93] for successfully climbing the specific level. As a precondition to climb on capability maturity level $i+1$, all goals of the KPAs related to level i must have been achieved in addition to the ones of level $i+1$. While its authors [PCCW93b] claim to have defined an "ordinal scale for measuring process maturity and evaluating process capability" it is vehemently challenged i. e. by Pfleeger et al. [PJCK97]

SPI approach
Being based on the cognition that evolutionary and systematic process improvement steps, one at a time, are more likely to lead to success than revolutionary process innovations, the *staged* CMM framework assigns sets of process areas to five distinct levels of organizational process maturity. Following the prescribed order of the process areas according to maturity levels aids in the prioritization of efforts. Moreover, process evolution and/or improvement towards the next higher maturity level is achieved, when all practices at lower levels are enacted and hence the process goals are fulfilled. [Hum88] This is an excellent member of the benchmarking-based family of SPI models.

B.3 ISO/IEC Standard 9001:2000

In 1987, when Humphrey published his process maturity framework based on best software engineering practices in a first technical report [Hum87] and a rating scheme in a second one [Hum87] that should relate the answers provided by the first one to 101 questions, ISO and IEC adopted this method as ISO/IEC Standard 9001 *Quality Systems — Model for Quality Assurance in Design/Development, Production, Installation and Servicing*. [BD93] Hence, this standard is in some way also connected with the DoD. [RHGH97] With the concern of quality and process management applicable to all industrial sectors, ISO/IEC Standard 9001 describes minimal characteristics of a

quality system [Pau95], which represented basic business requirements imposed by governmental contractors in the European Union. Then, in 1991, the companion guide ISO/IEC Standard 9000-3 *Guidelines for the Application of ISO 9001 to the Development, Supply, and Maintenance of Software* was published to facilitate the translation of ISO/IEC Standard 9001 in software engineering settings. [Coa94a] In the course of time, as illustrated in figure B.2, a related quality standards family emerged under the generic term ISO/IEC 9000. [SMH96] But most often software development organizations are solely urged to observe ISO/IEC Standard 9004-1 to establish a quality management system and either ISO/IEC Standard 9000-3 or TickIT's [Tic92] [Gib98] interpretation of ISO/IEC Standard 9001.

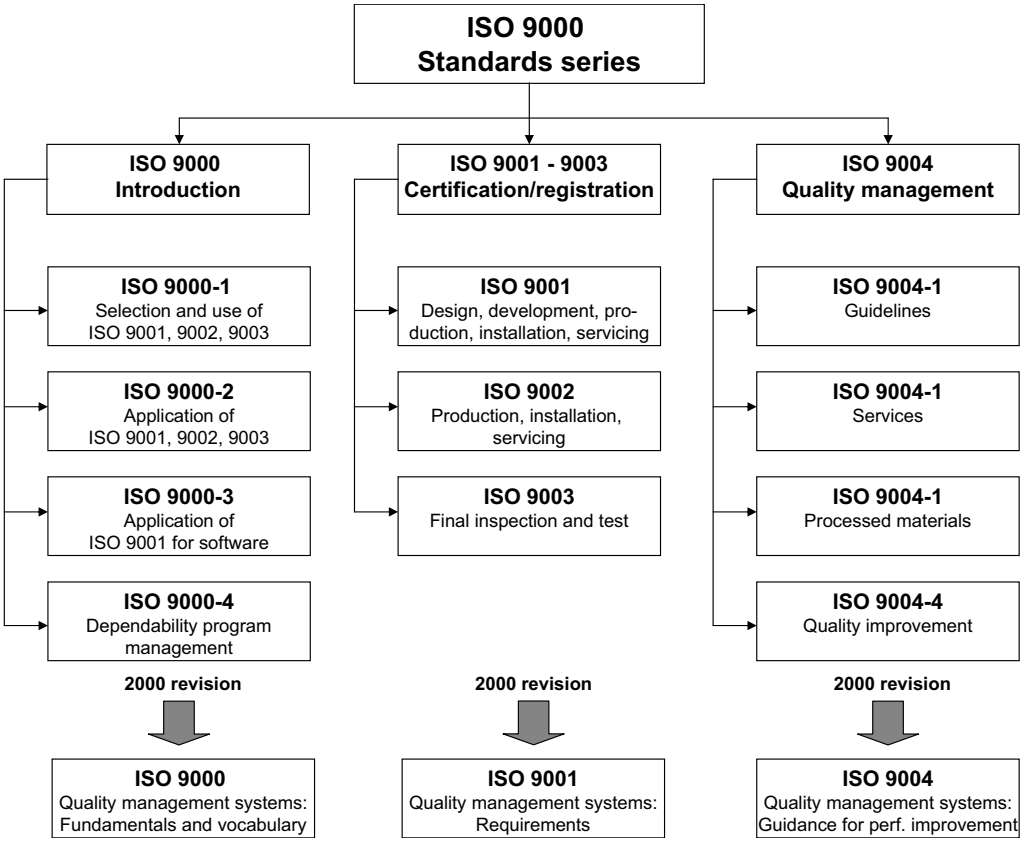

Figure B.2: Structure of ISO/IEC 9000 Standards series before and after 2000 revision

However, ISO/IEC Standard 9001 as amended and promulgated in 1994 has been criticized [Coa94a] for not really promoting the TQM notion of continuous improvement and for omitting factors contributing to quality like excellent leadership, human resources, marketing, and process engineering. Contrary to this argumentation, experiences [HJPH98] show that certification against ISO/IEC Standard 9001 has a SPI spin-off effect in related engineering areas. Indeed, in combination with other issues [Wal99] ISO and IEC decided to promulgate a major, more process-oriented revision in the year 2000. Simultaneously, the comprised standards were reorganized and joined together forming ISO/IEC Standards 9000, 9001, and 9004. In 2004, the ISO/IEC Standard 90003 has been published as concretion of Standard 9001 for the software engineering sector building a bridge to other IT-related *de jure* and *de facto* standards. Until December 2005, nearly 800,000 organizations have been examined for certification against ISO/IEC Standard 9001:2000 in 161 different countries with more and more buoyancy. [ISO05]

Process modeling approach
As representative of strictly prescriptive process models, what is indicated by numerous 'shall' statements [MS03], the rejuvenated ISO/IEC Standard 9001:2000 covers merely three elements of the process system taxonomy: Apart from general requirements, 21 processes relative to five process categories are communicated by means of clauses that represent the practices. [Sch03] In order to model all perspectives in all domains but not at the project domain, the model does not use a real PDL. The processes for implementing a quality management system are rather specified in natural language.

SPA approach
An appraisal of compliance with the requirements of the ISO/IEC Standard 9001:2000, is basically conducted via so-called certifications and/or registrations. [Coa94a] ISO/IEC do not certify any organization itself; instead many countries have accredited certification bodies (registrars) that perform the ISO certification process for applying organizations with costs. In doing so, the registrars and/or auditors themselves have to observe the ISO/IEC Standard 19011:2002 *Guidelines for quality and/or environmental management systems auditing*. To obtain an ISO/IEC 9001:2000 certification, organizations must *apply* to a national ISO registrar. Then, a rough on-site *pre-assessment* by the registrar of the organization predates the *document review and adequacy audits*, where the registrar can impose conditions to the management ('action requests' or 'non-compliances') to be remedied until the final *compliance audit*. After all, the registrar awards the compliance certificate once all requirements have been successfully met. However, the awarded and commonly treasured compliance certificate is only valid until the next surveillance visit of the registrar.

Process maturity model
The practice performance scale of the ISO/IEC Standard 9001:2000 falls back on the appraisal of the compliance with the standard's requirements and offers ratings of either 'Satisfied' or 'Not satisfied'. Despite the general trend, the standard does not provide a process maturity scale with multiple stages but offers the binary 'Fail' or 'Pass' decision at the organizational scope.

Process maturity determination method
Requiring full compliance, that is 100% positive answers for all requirements accommodated in pertinent questions, the process maturity determination model seems to be very straightforward but strict for the purpose of certification.

SPI approach
The alignment of the standard's processes as illustrated in figure B.3 on the lines of Shewhart's PDSA cycle [She31] and/or key elements of the TQM philosophy [KBS94] such as customer focus, the human side of quality, data, measurement, and analyses enables process-oriented continuous improvement of the quality management system. The rather analysis than real benchmarking approach fosters final quality targets of customer satisfaction, increased productivity, and short time-to-market periods.

B.4 BOOTSTRAP v2.22

To all intents and purposes, the mission of the BOOTSTRAP project under the ESPRIT was to fertilize the ground for the introduction of good software engineering practices in Europe by means of process assessment. [KKM+94] Predating the European System and Software Initiative (ESSI), a technology transfer program for European producers and users of software proposed by the Commission of European Countries,

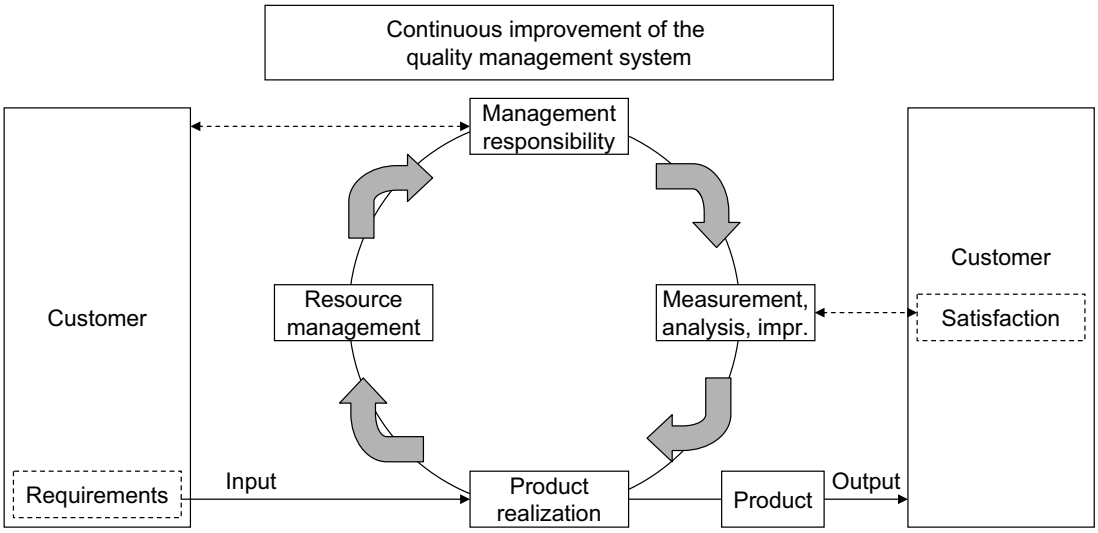

Figure B.3: Selected ISO/IEC Standard 9001:2000 processes in the context of TQM (adapted from the standard)

the BOOTSTRAP project had its phase-out in February 1993 with methodology version 2.22. [KB94] Despite the expiration of official support, the project was carried on by the homonymous consortium, the BOOTSTRAP Institute, which recently released version 3.0. [Kuv99] Being aware of the fact that SPA should predate SPI, the initial BOOTSTRAP methodology availed itself of the SEI's CMM together with guidelines from ISO/IEC Standard 9000-3:1987 and a classical software engineering standard PSS-05 [EC91] of ESA. [HJPH98] Between 1991 and 1997 over 100 assessments [SEL97] have been performed including major European companies like the German Robert Bosch GmbH or Siemens AG and its affiliated business divisions. [KM94] Because in the following years other methods could gain ground, the BOOTSTRAP methodology has become less important i. e. for that company.

Process modeling approach

In search of a "reasonable approach for referring any company-specific processes to a single model" [KB94], the authors of BOOTSTRAP committed themselves to the empirical, descriptive ESA software engineering standard PSS-05 [EC91], which was also adopted outside ESA by companies like General Motors or Ford [EFF+99] and admired for its tolerable level of detail. [SEL97] The process system taxonomy for the BOOTSTRAP process system turns out to provide the full range of elements: The process system is split into the two process subsystems of production and management that are organized by six categories, called key process clusters for 21 key process attributes (processes). In doing so, PSS-05 covers all modeling perspectives and describes processes of the complete life-cycle (either waterfall, incremental, or evolutionary delivery) by some 201 mandatory practices. [JMF97] Apart from mandatory practices, also recommended and guiding ones are given. Based on ESA's PSS-05 process model, BOOTSTRAP's authors modeled the goal of process quality at organizational or project level by providing certain key process attributes and/or key process clusters according to the three dimensions of organization, methodology, and technology. This attribute-based *quality attribute hierarchy* [HMK+94] is the foundation of the questionnaires and illustrated in figure B.4. Again, the process model is provided in natural language.

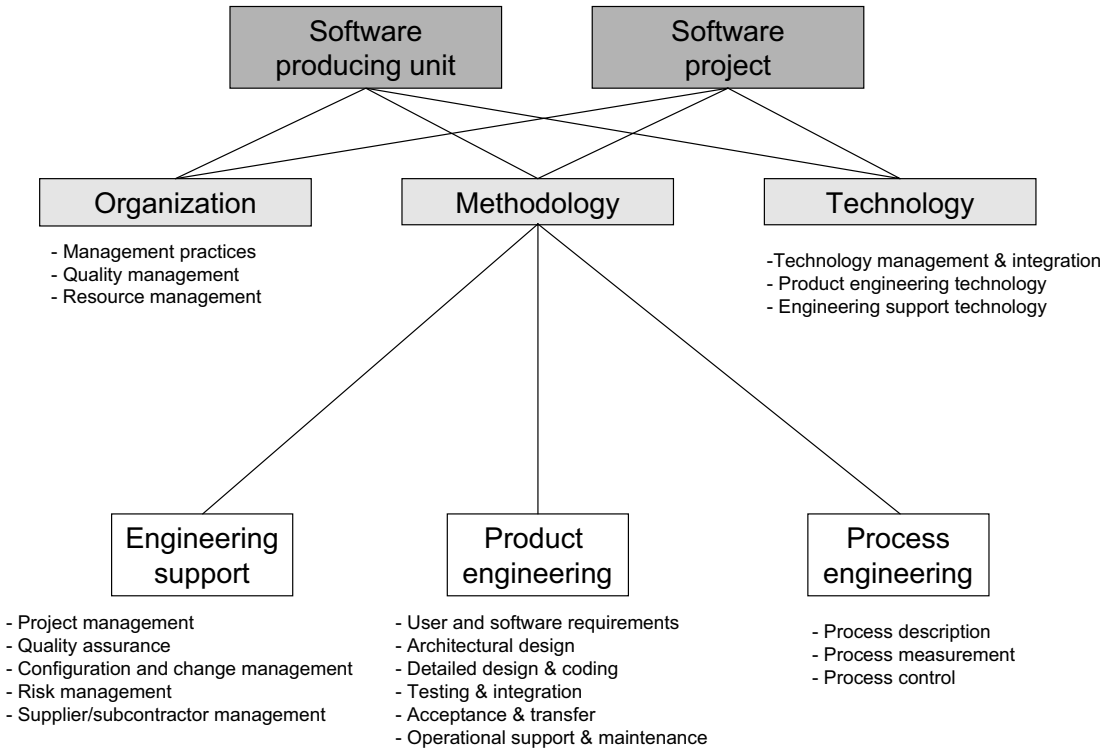

Figure B.4: BOOTSTRAP breakdown of key process attributes and clusters (adapted from [KB94, p. 123])

SPA approach
Imposed by the opinion that self-assessments do not share the same effectiveness as assisted assessments, the BOOTSTRAP methodology prescribes assessments by especially trained and experienced assessors and/or licensed consultants behaving with integrity. [KB94] [HJPH98] Because assessments can be conducted at organizational ('global site assessment') and project ('project assessment') level with the aid of respective questionnaires, the methodology is suited best for small and medium Software Producing Units (SPUs). [Kuv93] During the assessment activities several briefings predate guided interviews and document reviews, in which the respective questionnaires are completed by the assessor team with a minimum manning of two consultants, on-site. [HJPH98] To each of the 21 processes, a set of questions is devoted. [Kuv93]

Process maturity model
For answering the pertinent attribute-related questions according to the process maturity model binary decisions have been avoided by scaling the answers to three positive adjectives ('Weak', 'Fair', 'Extensive') of different severity, one neutral ('Non-applicable'), and one negative ('Absent') adjectives. [KB94] In a further contrast to its paragon, the SW-CMM, the strict sequence of questions for processes according to a certain capability maturity level has been suspended by also examining processes on all levels. [HMK+94] However, the process maturity scale of the SW-CMM has been adopted to fulfill the method's design requirements for the addressing the process and organizational scopes in a manner similar to the SW-CMM.

Process maturity determination method
Because the process maturity determination method for an assessment's results should be objective and follow clear-cut rules, a standardized algorithm compatible with the

SEI model was produced. Taking care of the quartiles, the algorithm labels an assessed company or project with a maturity level on the basis of the questionnaire's answers to each key process attribute and/or key process cluster as described in figure B.5. [KM94], if the overall score is within the maturity level's preset thresholds. [Kuv93] Because the number of questions decreases as the maturity levels increases, a dynamic step scale defines the percentage distances between the respective levels. [HMK+94] Moreover, with the aid of the BOOTSTRAP methodology absolute and relative *capability profiles* can be generated to pinpoint attribute-related process strengths and weaknesses that impact the maturity level and/or benchmark against competitor's overall mean score profiles stored in the BOOTSTRAP database. [SEL97]

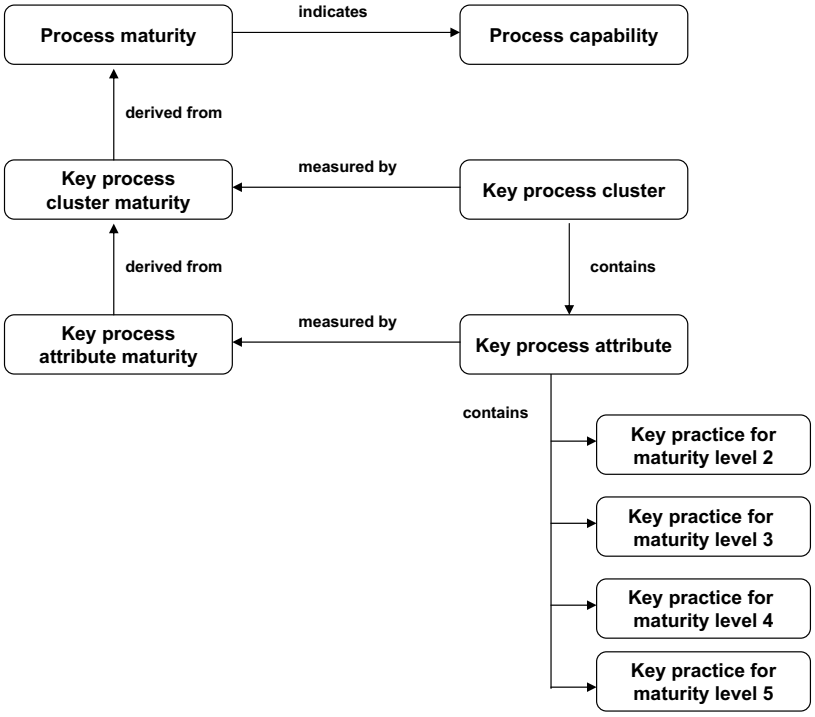

Figure B.5: Attribute-based structure of BOOTSTRAP (adapted from [HMK+94, p. 27])

SPI approach
In order to act on reliable, complete, and consistent assessment results, all interviewees agree on, the assessment's results have to be verified. Afterwards, based on the maturity level, model target profiles for SPI have to be found, that will most probably be one-step improvements for each process category following the *staged* improvement approach that has been adopted from the SW-CMM. Predating the preparation of a virtual improvement schedule, priorities in terms of most urgent issues are set. This SPI model is again a representative of the benchmarking approach.

B.5 TRILLIUM v3.0

The TRILLIUM model, with the most recent version 3.0 of 1995, has been developed since 1991 by a joint-venture project of Bell Canada, Northern Telecom, and Bell-Northern Research to minimize acquisition risks of predominantly embedded-systems software in the telecommunications sector on the base of an assessment of a supplier's development process maturity. [Coa94b] [May96] Despite being strongly influenced by the SW-CMM and other cutting-edge models and standards, its authors placed emphasis on a product rather than pure software perspective and geared the model towards

customer satisfaction, which is one of the fundamental ideas of TQM. Besides auditing, TRILLIUM is also appropriate as a sound foundation for vendors' continuous improvement programs of product development and support capabilities. [AC95] In industrial applications at Bell Sygma Telecom Solutions the model could show its value and viability. [CMWH96] Additionally, starting from the Anglophone TRILLIUM model, the cooperative project *Camélia* funded by the French government and the Canadian province Québec translated and optimized the model for assessing Management Information Systems (MISs). [Apr05]

Process modeling approach
On the whole, the practices of the empirical, descriptive TRILLIUM process model have been derived from a 'benchmarking exercise' [AC95] [Coa95] which availed itself mainly of SEI's SW-CMM but also on ISO/IEC Standard 9001:1994 together with the associated guidelines of ISO/IEC Standard 9003-1994, proprietary Bellcore standards, MBNQA criteria of 1995, and 1993 version of software engineering standards of IEEE and IEC Standard 300:1984. Additional practices not covered by the above listed material were incorporated, as well. In the taxonomy of TRILLIUM's process system, there are eight process categories, the capability areas, that contain one or more and altogether 28 processes, called road maps, of which each contains a set of related practices. All in all, 508 practices constitute the extent of the model that addresses all perspectives of the organizational, development, and management domain by the process model in natural language.

SPA approach
Although the documentation of the TRILLIUM model [Coa94b] remains relatively silent about conducting appraisals [Jok01], it provides some general information on assessment activities to be conducted by assessors having been trained either on the TickIT [Tic92] interpretation of ISO/IEC 9001-1987, ISO/IEC Standard 10011-2 that has been replaced by ISO/IEC Standard 19011:2002 in the meanwhile, early assessment scheme of CMM [OHK89], or MBNQA criteria. Withal, three distinct types of assessments are proposed: Supplier capability evaluations, joint-assessments between customer and vendor, and self-assessments of the supplier. [AC95] Due to the explicit closeness of the TRILLIUM model to its parent the CMM, the process maturity model as well as the process maturity determination model are relatively congruent with their respective archetypes of CMM. For reasons of completeness the hierarchical maturity model's architecture is presented in figure B.6.

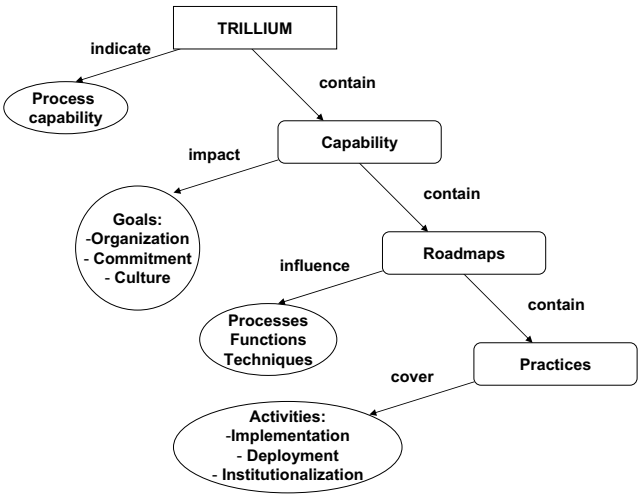

Figure B.6: Hierarchical architecture of the TRILLIUM model (adapted from [Coa95])

SPI approach
Again, the improvement approach immanent in the TRILLIUM model is that of Shewhart's [She31] PDSA cycle connected with the ideas of TQM. [Ish85] The model suggests to have at least one representative of a customer on-site to interact with the supplier's improvement program to communicate and enforce issues important for his satisfaction. [AC95] [Coa95] Here, the yardstick for benchmarking is represented by the best practices as manifested in the TRILLIUM model.

B.6 CMMI Framework v1.1

After the initial promulgation of SEI's framework in 1987 that evolved to the CMM, a myriad of CMMs has been developed for specific disciplines. [Kne06] Even though the specific models proved to be useful when applied isolated, the usage of multiple models in parallel bore problems that limited organizations in their successes with SPI initiatives, let alone additional cost and effort for training and appraisals. [CKS03] Motivated by that obstacle, the mission of the Capability Maturity Model Integration (CMMI) project was to combine the latest versions of three main CMMs of the SEI, namely SW-CMM, SECM, IPD-CMM, and to integrate them to their designated successor CMMI. [SEI02b] [SEI02a] A premise in doing so was to maintain consistency and compatibility with the then-emerging ISO/IEC Standard 15504 Technical Report, on which is dwelled on in section 3.3. Mirroring expertise knowledge from the bodies of knowledge of Systems Engineering (SE), Software Engineering (SW), Integrated Product and Process Development (IPPD), and Supplier Sourcing (SS), the combinations most useful to the interested organization like CMMI-SE/SW or CMMI-SE/SW/IPPD/SS can be instantiated.

Although the CMMI Framework had to struggle because the SEI or rather the decision makers at DoD acted ungratefully to the contributors of the parent models, "... the users of the model are in fact utilizing the variety of choices in representation, model scope, and whether to do a formal rating." [Zub03] After debunking cut and dried opinions against CMM's successor [Hei04], successful application of the framework can be found multiply in the literature, with increasing frequency. [PA03] [EL06] Because the up-to-date version 1.2 of the CMMI Framework for development [SEI06] has been released as recently as in August 2006, it is out of scope here.

Process modeling approach
The CMMI Framework integrates the above mentioned empirical, descriptive process models of the SEI. Because CMMI's authors recognized different, either *staged* or *continuous* approaches to SPA/SPI being immanent in the parent models, they provide these two representations of the underlying process model. [CKS03] Dependent on the form of the representation the taxonomy for the CMMI process system differs with the staged representation being similar to the well-known SW-CMM: In the staged representation, there are five maturity levels organizing 25 process areas comprising 185 specific and twelve generic practices; in the continuous form, the basic structure is slightly modified while the basic information is retained. There are six levels of process capability that span on 25 process areas, in which 189 specific and 17 generic practices are covered.

For modeling the entire spectrum of process perspectives with respect to the organizational, development, and management domain, the authors of the CMMI combined the parenting models into natural language representation.

SPA approach
To specify the SPA approach for the CMMI Framework version 1.1 two major documents have been developed over time:

- The Appraisal Requirements for CMMI (ARC) [SEI01a] setting out requirements for three different classes of appraisal methods with decreasing amount of rigor and significance, and
- The SCAMPI Method Description Document (MDD) [SEI01b] comprising the methodology for the only approved class A appraisal method. [KJ03]

The ARC distinguishes between *assessments*, mainly performed by personnel internal to the organization with the purpose of assessing SPI efforts, *process appraisals*, conducted by a trained team of professionals on the base of an appraisal reference model to determine strengths and weaknesses, and *evaluations*. The latter are appraisals, in which external groups examine an organization's process and decide about future performance of business. According to these three different appraisal types, the three classes A, B, C of appraisals with decreasing amount of rigor are defined by the ARC, as well. [Min02]

Until recently, the SCAMPI, which is based on experiences with methods known from the SW-CMM such as CBA-IPI or SCE, was the only class A method, that has been approved by the SEI for evaluations and for providing ratings of the organization. Besides SPI, the use of the SCAMPI method is, however, also encouraged for supplier selection and process monitoring with internal appraisal teams. [KJ03] The most rigorous and expensive SCAMPI evaluation must be led by an SEI-authorized Lead Appraiser who is supported by a full range of work aids, maturity questionnaires, and the like that altogether form the SCAMPI Product Suite. Several steps like gathering and reviewing documentation, conducting interviews, discovering strengths and weaknesses, and presenting the findings have are run through in the course of an evaluation.

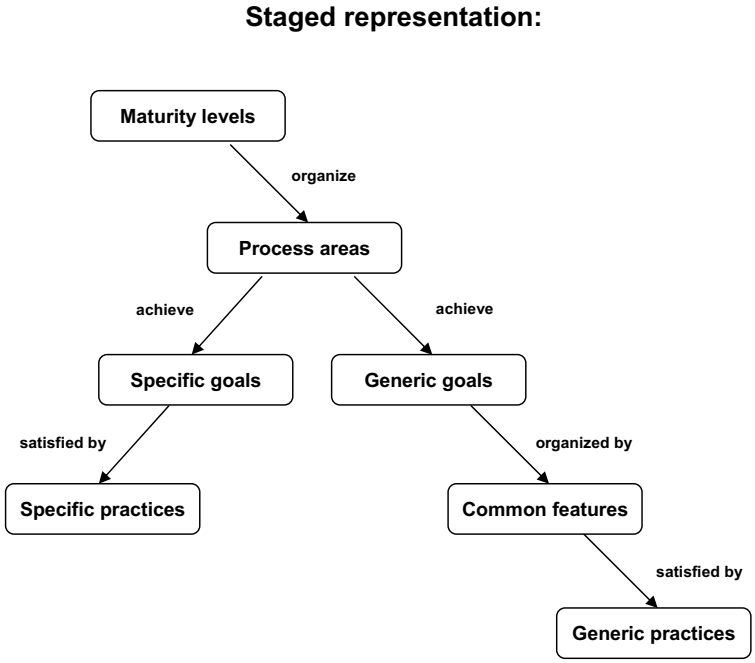

Figure B.7: Model components in the staged representation of the CMMI Framework v1.1 (adapted from [SEI02b, p. 10])

Process maturity model
As already indicated, two representations, that is the staged and the continuous, dominate the process maturity model of the CMMI Framework version 1.1 as depicted in figures B.7 and B.8, respectively.

Continuous representation:

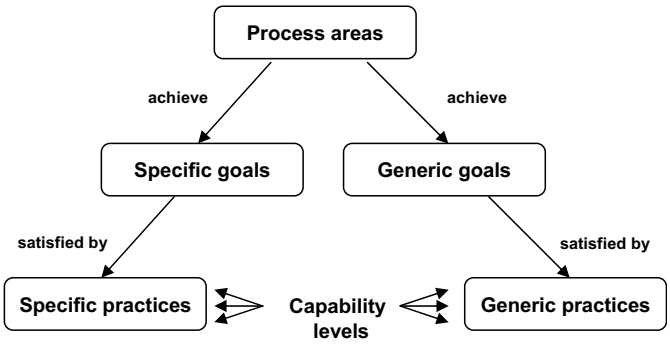

Figure B.8: Model components in the continuous representation of the CMMI Framework v1.1 (adapted from [SEI02a, p. 12])

In doing so, two different paths for improvement are offered:

- First, an organization could avail itself of the staged representation. Then it could improve its *maturity* by addressing predefined, successive sets of process areas that encounter at each *maturity level*.
- Second, an organization might choose the continuous representation to select one or more specific process areas out the four groups: 'Process Management', 'Project Management', 'Engineering', 'Support'). Consequently, an incremental improvement of the chosen process area's *capabilities* can take place along *capability levels* towards a targeted level.*

This leads to the following definitions as per [CKS03, p. 75] that are valid in the context of the CMMI Framework v1.1:

> **Definition B.1:** "*Maturity levels*, which belong to a staged representation, apply to an organization's process improvement achievement across multiple process areas."
>
> **Definition B.2:** "*Capability levels*, which belong to a continuous representation, apply to an organization's process improvement achievement in individual process areas."

In the staged form, five levels of overall process maturity organize key process areas which are performed to achieve goals. While the goals specific to the process areas shall be satisfied by performing specific practices, two generic goals, that are of importance to any process area, are organized by four common features ('Commitment to Perform', 'Ability to Perform', 'Directing Implementation', and 'Verifying Implementation'), which shall be satisfied by a number of generic practices. It is important to note that the model offers a different stringency of interpretation: It requires goals to be achieved, only expects practices to be performed, and provides elements like sub-practices to inform the user. [ACT03]

*While the selection of the process area(s) together with the target capability is called 'target profile', the actually tracked progress is reflected by an 'achievement profile'.

In the continuous form, the sophistication of performed practices is indicated by different levels of capability for a singular process area. Both, specific and generic goals (each for one capability level above zero) have to be achieved with the aid of a quantity of specific and generic practices. There are *base practices*, that are situated at entry capability level one and *advanced practices* showing more sophistication and rigor. Withal, a capability level higher than one is seen as the degree to which advanced practices are performed in the organization. [KJ03]

For the purpose of the assessment the process maturity model does not differ between the two representations of the model but between required, expected, and informative model components during the evaluation: The appraiser has to judge (rate) on the base of the available materials for any specific and generic goals of the process areas under evaluation, whether they are satisfied. Therefore, he uses a four-point practice performance scale, in which the goals are indicated to be either 'Satisfied', 'Unsatisfied', 'Not applicable', or 'Not rated'. [SEI01b] Afterwards, relative to the kind of representation the process maturity scale is structured: To rate an organization's overall process maturity the range of five maturity levels is available: 1 – Initial, 2 – Managed, 3 – Defined, 4 – Quantitatively managed, 5 – Optimizing. Over and above, it is also possible to rate the capability of a single process out of the organization's software engineering process system on a scale of six capability levels: 0 – Incomplete, 1 – Performed, 2 – Managed, 3 – Defined, 4 – Quantitatively managed, 5 – Optimizing. In doing so, practices, processes, projects, and even organizations can be rated using those scales by assigning in a 2-D manner, either a capability level or a maturity level.

Process maturity determination method
Contrary to the process maturity model, the process maturity determination method does distinguish between the continuous and staged representation of the model: For the continuous representation, a certain process area is labeled with certain *capability level*, when all specific and the respective generic goal of the process area for the capability level are rated with the label *satisfied* together with the cumulative specific and generic goals of previous capability levels. Afterwards, a capability profile illustrating the ratings of all process areas appraised can be produced. For the staged representation a certain *maturity level* is given to the entire organization. The maturity level can only be awarded to the organization, when all capability levels of the process areas organized by the certain maturity level as well as all process areas of the previous maturity levels have been rated with the label *satisfied*. [SEI01b]

SPI approach
Owing to the fact that the CMMI project team demonstrated longsightedness when providing both, a staged and a continuous representation of the process model, two accepted ways for benchmarking-based SPI are supported [CKS03]:

When an organization is in need of a systematic and structured way doing improvements one step at a time, the staged representation offers cumulative, predefined sets of process areas for each maturity level. By prescribing the order of implementing each process area relative to the maturity level it offers a precise path for improvement. [ACT03] In contrast, the continuous representation can be considered, when a more flexible and narrowed down approach to SPI is desired. Then, an organization might want to choose a process area, which has been identified as a trouble spot for instance by an appraisal, and can try to improve relative to it. [KJ03]

B.7 Other models

As reaction to the mainstream models for SPA/SPI derivatives have been developed such as the Personal Software Process (PSP) [Hum96] [Hum05a], the Team Software Process (TSP) [Hum99] [Hum05b], or eXtreme Programming. [BA04a] Since the spirit of the age was sympathetic to the basic idea underlying those models, a myriad of niche models geared towards special processes has been developed over the time, too. While the author of this thesis provides a more comprehensive list in a recent technical report together with Dumke et. al [DBB+06a], the probably best known are listed next:

- Testing Maturity Model [BSC96]
- Formal Specifications Maturity Model [FV97]
- IT Service Capability Maturity Model [Nie00]
- Corrective Maintenance Maturity Model [KM01]
- Documentation Maturity Model [HT03]
- E-Learning Maturity Model [MM04]
- Software Maintenance Maturity Model [Apr05]
- Requirements Engineering Process Maturity Model [SR05]
- Requirements Process Improvement Model [BHR05]
- SPI Implementation Maturity Model [NWZ05]
- Simulation Validation Process Maturity Model [HY05]

The Six Sigma approach
Sigma (σ) stands for standard deviation of anything. The Six Sigma approach in the software development field was considered an interval (six: three at both sides) which keeps a 99.9 percent correctness as absence of any defects. [Tay02] The following table shows the defect percentage depending upon the different sigma levels.

Sigma level	Percent correct	#Defects per million opportunities
3	93.3193	66807
4	99.3790	6210
5	99.9767	233
6	99.99966	3.4

Table B.1: Characteristics of different sigma levels

The kernel process of the Six Sigma approach includes/uses five phases referred to as the Define – Measure – Analyze – Improve – Control (DMAIC) model, which means:

1. *Define* the problem and identify what is important (define the problem, form a team, establish a project charter, develop a project plan, identify the customers, identify key outputs, identify and prioritize customer requirements, document the current process).
2. *Measure* the current process (determine what to measure, conduct the measurements, calculate the current sigma level, determine the process capability, benchmark the process leaders).

3. *Analyze* what is wrong and potential solutions (determine what causes the variation, brainstorm ideas for process improvements, determine which improvements would have the greatest impact on meeting customer requirements, develop a proposed process map, and assess the risk associated with the revised process).

4. *Improve* the process by implementing solutions (gain approval for the proposed changes, finalize the implementation plan, implement the approved changes).

5. *Control* the improved process by ensuring that the changes are sustained (establish key metrics, develop the control strategy, celebrate and communicate success, implement the control plan, measure and communicate improvements).

The general aspects of the Six Sigma approach are shown in the following figure:

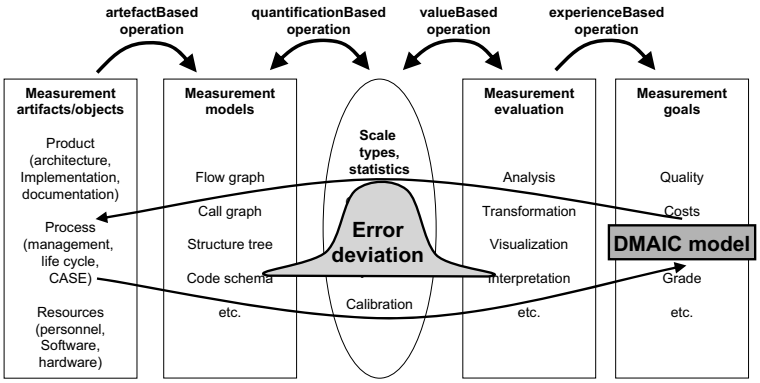

Figure B.9: Basic characteristics of the Six Sigma approach (adapted from Dumke [Dum05])

Furthermore, the Six Sigma approach is available for the traditional software development life cycle, legacy systems, package software implementation, and outsourcing. [Tay02]

The ITIL approach

The IT infrastructure library (ITIL) is a set of documents that are used to aid in the implementation of a framework for IT service management. [ITI06] This framework characterizes how service management is applied within an organization. ITIL was originally created by an UK Government agency; it is now being adopted and used across the world as the *de facto* standard for best practice in the provision of IT services.

ITIL is organized into a series of sets as a best practice approach, which are divided into eight main areas:

1. *Service Support* is the practice of those disciplines that enable IT services to be provided effectively (service-desk, incident management, problem management, change management, configuration management, release management).

2. *Service Delivery* covers the management of the IT services itself (service level management, financial management for IT services, capacity management, service continuity management, availability management).

3. *Security Management* considers the installation and realization of a security level for the IT environment (trust, integrity, availability, customer requirements, risk analysis, authority, and authenticity).

4. *Infrastructure Management* describes four management areas: design and planning, deployment, operations, technical support.

5. *Application Management* describes the service life cycle as requirements - design - build- deploy - operate - optimize.

6. *Planning to Implement Service Management* defines a guide in order to deploy the ITIL approach in a concrete IT environment.

7. The *Business Perspective* describes the relationships of the IT to the customers and users.

8. *Software Asset Management* defines the processes and the life cycles for managing the software assets.

The following triangle characterizes the different relationships between the service management standards and ITIL:

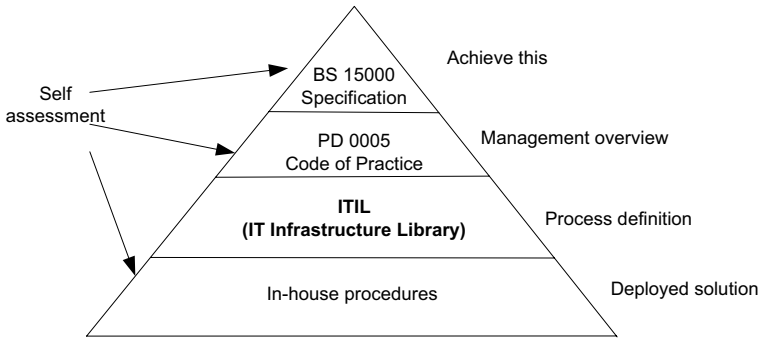

Figure B.10: The relationship between the service standards and ITIL

Here, BS 15000 is the service management standard, ISO/IEC Standard 20000 describes the specification for service management, and PD 0005 stands for code of practice for the IT service management (ITSM). Usually, the implementation of the ITIL approach is supported by any *ITIL toolkit*.

B.8 Typology and conclusion

Typology
As a result of a comparative study among prevalent capability maturity models, Fraser et al. [FMG02] summarize their findings in a brilliant typology of types of models:

1. *Maturity grids* containing textual descriptions of the pertinent practices at the distinct levels of maturity.

2. *Hybrids and Likert-like questionnaires* embody the 'good practices' in the questions for which respondents have to score the level of performance using e. g. an ordinal scale from 1 to n.

3. *CMM-like models* stand for the sort models that bear a formal and complex but concerted architecture of distinct process elements. (Note: All of the models briefly visited in this chapter belong to that type of models!)

Conclusion
Examining the constituents of cutting-edge capability maturity models and/or SPA/SPI methods, researchers such as El-Emam et al. [EG99] or Kuvaja [Kuv99] found out that commonly the main features are (1) a process reference model which together with (2) principles for scoring and determining of the current process performance in terms of capability levels as well as result presentation form (3) an assessment method. After all, (4) process improvement guidelines top off the models. Despite the diversity of special software engineering topics covered by those models, Fraser et al. [FMG02] observe that "they share the common property of defining a number of dimensions or process areas at several discrete stages or levels of maturity, with a description characteristic performance at various levels of granularity."