Framework for a Service-oriented Measurement Infrastructure

Dissertation

zur Erlangung des akademischen Grades

Doktoringenieur (Dr.-Ing.)

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

von: Dipl.-Inform. Martin Kunz
geb. am 10.03.1980 in Burg (b. Magdeburg), Deutschland

Gutachter:

Prof. Dr.-Ing. habil. Reiner Dumke

Prof. Dr.-Ing. habil. Georg Paul

Prof. Dr. Juan José Cuadrado-Gallego

Magdeburg, den 5. Juni 2009

Kunz, Martin
Framework for a Service-oriented
Measurement Infrastructure
Dissertation,
Otto-von-Guericke University of Magdeburg,
2009.

Acknowledgement

The successful completion of this research work would not have been possible without the support, guidance, and cooperation of several fellows.

At first, I would like to gratefully acknowledge the supervision of Prof. Dr. Reiner Dumke during this work. Especially his continued encouragement, invaluable suggestions, and the valuable comments he has given over the years significantly contributed to this thesis.

I am also very grateful to Prof. Dr. Juan-José Cuadrado-Gallego and Prof. Dr. Georg Paul for their efforts in reviewing and providing their expert opinions on the thesis at hand.

Special thanks go to Dr. René Braungarten, Daniel Reitz, and Niko Zenker, for their friendship, encouragement, hard questions, and helpful remarks over the years. The joint research, conference travels, and fruitful discussions resulted in deep friendship and mutual advice.

Furthermore, I am deeply indebted to my former colleagues at the Otto-von-Guericke University of Magdeburg that have provided the environment for sharing their experiences about the problem issues involved as well as participated in stimulating team exercises developing solutions to the identified problems. I would specially like to thank my fellows Prof. Dr. Andreas Schmietendorf, Dr. Fritz Zbrog, Dr. Steffen Mencke, Dr. Dmytro Rud, Dagmar Dörge, and Ayaaz Farooq for their support and insights.

Moreover, I would like to thank Prof. Dr. Alain Abran and Prof. Dr. Alain April from the ÉTS at Montréal, Dr. Luigi Buglione from Engineering.IT, Dr. Marek Leszak from AlcatelLucent, Dr. Evgeni Dimitrov from T-Systems, Dr. Yoshiki Mitani from NAIST at Tokyo and Harry Sneed from the University of Regensburg for their cooperation and advice.

I am also thankful to Scientific Toolworks and Telelogic for providing evaluation copies of their tools which was essential for analyzing them. Additionally, I want to thank the ISBSG for an evaluation copy of their benchmarking suite and T-Systems for detailed insights into their metrics database system.

Finally, my very special thanks belong to my family for their warm support and strength all these years.

A special thought is devoted to my father for a never-ending support.

Burg, April 2009

Martin Kunz

Abstract

The increasing economic relevance of software measurement for organizations cannot be neglected. But issues like complexity and missing traceability of measurement processes constitute the need for direction and measurement tool support.

Unfortunately, the area of software measurement tools is dominated by inflexible, monolithic, and self-contained tools. This situation aggravates a process comprehensive solution and results in n unsatisfying situation regarding corporate measurement programs.

Due to manifold advantages of high-flexible infrastructures compared to monolithic products a lot of initiatives propose approaches for the integration of single components (e.g. services).

Having analyzed the SOA-capability of existing measurement tools this thesis introduces a framework for creating a measurement infrastructure by means of a service-oriented architecture.

Beyond the presentation of different components to implement the infrastructure the specific relevance of software measurement databases is addressed by the design of a service-oriented measurement database.

Beside the functional characteristics the quality of developed architectures is of substantial interest for the success of systems integration in the long run. Therefore for a procedure for quality driven design of service-oriented architectures has been integrated into the framework.

Additionally, formal considerations of existing paradigms in comparison to the service-oriented approach constitute the reasonability of the presented research topic.

Table of contents

List of Figures

List of Tables

Table of Abbreviations

AOP	Aspect-Oriented Programming
ANSI	American National Standards Institute
AOSE	Agent-Oriented Software Engineering
API	Application Programming Interface
ASG	Allen Systems Group
BPMN	Business Process Modeling Notation
BPEL	Business Process Execution Language
CASE	Computer-Aided Software Engineering
CAME	Computer-Assisted Measurement and Evaluation
CBSE	Component-Based Software Engineering
CBO	Collaboration Between Objects
CFP	Cosmic Function Point
CMM	Capability Maturity Model
CMMI	Capability Maturity Model Integration
COCOMO	Constructive Cost Model
COM	Common Object MOdel
CORBA	Common Object Request Broker Architecture
COSMIC	The Common Software Measurement International
COTS	Commercials Off The Shelf
CSV	Comma Separated Values
CWM	Common Warehouse Metamodel
C&K	Chidamber & Kemerer
DACS	Data & Analysis Center for Software
DAML	DARPA Agent Markup Language

DBMS	Data-Base Management System
DBS	Data-Base System
DCE	Distributed Computing Environment
DDL	Data Definition Language
DML	Data Manipulating Language
DIT	Depth of Inheritance Tree
DPDS	DACS Productivity Center
DTD	Document Type Definition
EAI	Enterprise Application Integration
EBD	Event-Based Design
EF	Experience Factory
ERP	Enterprise Resource Planning
ETL	Extract Transform Load
FOD	Feature-Oriented Design
FP	Function Point
GQM	Goal-Question-Metric
GUI	Graphical User Interface
HTML	HyperText Markup Language
ICASE	Integrated Computer Aided Software Engineering
ICT	Information and Communication Technology
IDL	Interactive Data Language
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical & Electronics Engineers
IFPUG	International Function Point Users Group
IPD-CMM	Integrated Product Development CMM
IPSE	Integrated Project Support Environment
ISBSG	International Software Benchmarking Standard Group

ISO	International Standardization Organization
IT	Information Technology
J2EE	Java Platform, Enterprise Edition
KDSI	Kilo Delivered Source Instructions
LCOM	Lack of Cohesion in Methods
LOC	Lines of Code
MAS	Multi-Agent System
MP	Measurement Process
NASA	National Aeronautics & Space Administration
NOC	Number Of Children
ODMG	Object Data Management Group
OIM	Open Information Model
OLAP	Online Analytical Processing
OMG	Object Management Group
OMT	Object Modeling Technique
OOAD	Object-Oriented Analysis and Design
OOMO	Object-Oriented Measurement Ontology
OOSE	Object-Oriented Software Engineering
OWL	Ontology Web Language
PDF	Portable Document Formant
PNG	Portable Network Graphics
PSM	Practical Software Measurement
QIP	Quality Improvement Paradigm
QoS	Quality of Service
QuaD^2	Quality Driven Design
QMP	Quality Model
RFC	Response For a Class

ROI	Return On Investment
RTF	Rich Text Format
SANTA	Solution Architecture for N-Tier Applications
SD	Software Development Process
SDK	Software Development Kit
SDP	Software Development Project
SE	Software Engineering
SECM	Software Engineering Capability Model
SEMS	Software Engineering Measurement System
SLED	Software Lifecycle Empirical/Experience Database
SMDB	Software Measurement Data-Base
SML@b	Software Measurement Laboratory
SMP	Software Measurement Program
SOA	Service-Oriented Architecture
SOAP	Simple Object Access Protocol
SOMDB	Service-Oriented Measurement Database
SOSE	Service-Oriented Software Engineering
SP	Software Product
SPARC	Standards Planning and Requirements Committee
SPC	Statistical Process Control
SPICE	Software Process Improvement and Capability Determination
SQL	Structured Query Language
SR	Software Development Resources
TIM	Type Information Model
TLB	Type Libraries
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language

UREP	Universal Repository
URI	Unified Resource Identifier
W3C	Word Wide Web Committee
WBSE	Web-Based Software Engineering
WMC	Weighted Methods per Class
WS	Web Service
WS-CDL	Web Service Choreography Description Language
WSDL	Web Service Description Language
XML	eXtensible Markup Language
XSD	XML Schema

1. Introduction

1.1. Motivation

The influence of software in our daily life has been raised dramatically over the past decades. Software is an important part almost everywhere and used for various reasons from sending a greetings card via email to automatically control the flight of an airplane. Living without software is nearly unimaginable these days. These increased influence and the associated usage was made possible in particular through the rapid price decline of computer software and hardware in the past decades. This process was again accelerated through the globalization in the nineties.

From software producers point of view new challenges arises through this development: competitors have to respond very fast to the customer's demand for high quality and low priced products.

Through the incoherency of these requirements a huge risk for development projects arise and in many cases projects overrun time or budget or they fail achieving high quality.

To avoid failure of projects even under these requirements and to clinch project goals methods, procedures, and tools have to be evolved and have to be applied to support software development in every of its branches.

One approach to target the set of problems is to identify well-tested methods from other engineering disciplines which are faced with comparable challenges. A second one is to observe leading competitors which achiever better results than their fallen behinds.

In doing so, beyond minor factors like excellent technical staff or professional work climate, such leading organizations achieving their goals by applying a quantitative approaches to software development that has been successfully used over large number of years.

Therefore the measurement of the software development process, software development resources, and software work products can provide important support to overcome conflictive requirements within the software engineering discipline.

A setting of tasks arises in particular for software management with it sub disciplines project management and quality management, to make decisions on basis of certain measurement values and with the application of standardized methods.

Another motivation for the usage of software measurement in the scope of a software development project was already outlined by Mary Shaw in the early nineties. She identified that software measurement is required to evolve the software development to a professional engineering discipline where measurements are used to distinguish whether or not specified quality requirements have been achieved during the development process [Jones96].

Figure 1 shows the evolution of an engineering discipline where the lower line track the technology and the upper line represent the influence of production skills and scientific knowledge carry new capabilities to the engineering practice.

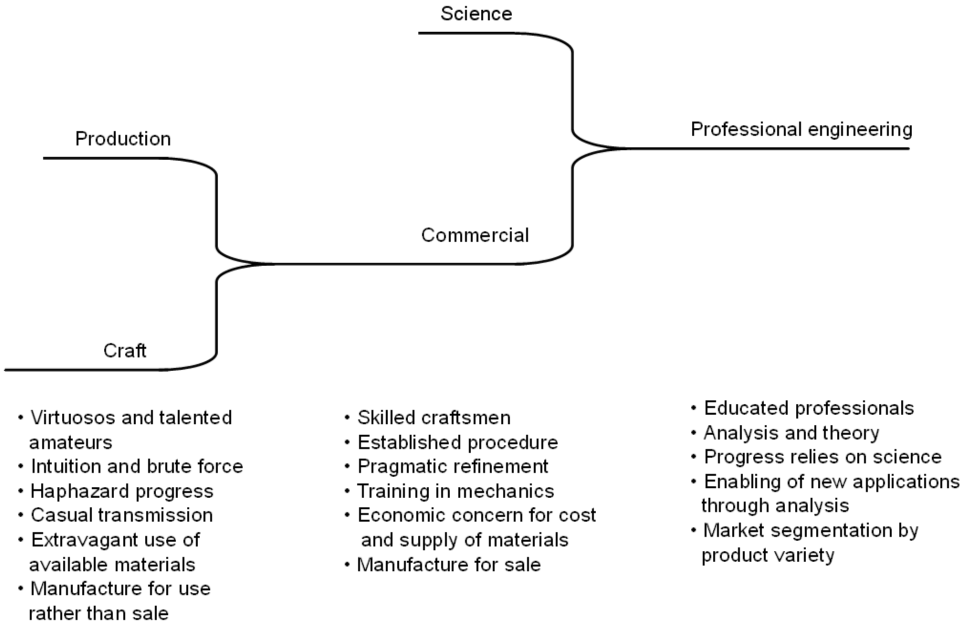

Science
Production
Professional engineering
Commercial
Craft
• Virtuosos and talented amateurs
• Intuition and brute force
• Haphazard progress
• Casual transmission
• Extravagant use of available materials
• Manufacture for use rather than sale
• Skilled craftsmen
• Established procedure
• Pragmatic refinement
• Training in mechanics
• Economic concern for cost and supply of materials
• Manufacture for sale
• Educated professionals
• Analysis and theory
• Progress relies on science
• Enabling of new applications through analysis
• Market segmentation by product variety

Figure 1: Evolution of an engineering discipline [Shaw90]

In comparison to other engineering disciplines the evolution software engineering is still in progress. Even though the term "software engineering" is used since 1968 [Shaw90] and the application of scientific analysis and standardized methods emerged the lack of mature tools, trustworthy experience, and the lack of established software measurement technology draws the picture that the path to professional engineering is still in progress [Boehm06]

The quantifiable approach of software measurement is backed up by IEEE's definition of software engineering [IEEE90]:

(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).

In a formal consideration according to [Dumke05] the entities comprised in software engineering (SE) are SE methods, supporting tools of Computer-Aided Software Engineering (CASE), system of measures, standards, the produced software system, derived experience which is shared in SE-communities:

$$\mathbf{SE} = (M_{SE}, R_{SE})$$

$$= (\{SE\text{-}Methods, CASE, SE\text{-}SystemOfMeasure\ , SE\text{-}Standards,$$

$$SE\text{-}SoftwareSystems, SE\text{-}Experience, SE\text{-}Communities\}, R_{SE}\,)$$

The realization of software engineering in order to conduct a specific development project leads to the definition of Software Engineering Management as the *"application of management activities – planning, coordinating, measuring, controlling, and reporting – to assure that the development and maintenance of software is systematic, disciplined, and quantified!"* [AbranMoore⁺04].

In order to support software management software measurement has been identified as an important assistance.

The interplay of software measurement and management phrases by the following citation: *"management without measurement […] suggests a lack of rigor, and measurement without management suggests a lack of purpose or context."* [AbranMoore⁺04]

1.2. Research Question

One on the major success factors for the applicability of software measurement is the overcoming of manual activities in software measurement. Therefore tools are used to automate as much activities as possible.
The dedicated task for this thesis is to investigate the lack of measurement maturity and measurement tool distribution.
The main research question that is investigated by the thesis is therefore:

How should an infrastructure for software measurement be arranged, to implement the software measurement process of an organization?

To address the primary research question the following questions are also investigated which simultaneously represent the scope of the thesis:

1. What are the special challenges for the implementation of software measurement programs?
2. Which basic software measurement approaches are significant for the software engineering discipline?
3. Which software measurement process should be realized by a measurement infrastructure?
4. What measurement process level can be observed in existing solutions?
5. Which elements should be part of a measurement infrastructure and how should an example be implemented?
6. How can quality requirements alongside with functional requirements quality requirements be achieved?

1.3. Research Methodology

The following section should give an overview about the used research methodology and steps which resulted in the thesis in hand.

Step 1 – Definition: consists the following activities

(1) Identification of a problem which has never before solved in industry or academia

(2) Documentation of the research question

(3) Identification and documentation of the methodology for the proposed solution.

Step 2 – Planning of the research project: consists a literature study to the state of the art of the two significant topics:

(1) Software measurement

(2) Software measurement programs

Step 3 – Design and implementation of the proposed solution

Step 4 – Interpretation and Evaluation of the proposed measurement infrastructure

Table 1: Step 1 - Definition

Motivation	**Objectives**	**Proposal**	**Research Stakeholders**
How should a measurement infrastructure be designed to automate the software measurement process?	Evaluate tools and architectures and support the enhancement.	A framework that contains components to realize a standardized software measurement process.	- measurement tool manufactures - software measurement tool users - researcher in software measurement

Table 2: Step2 - Planning

Stages of the project	**Inputs**	**Outputs**
Literature review of the detailed practices of software measurement programs	- literature review of software measurement - literature review of corporate measurement	- a technical report covering the state of the art in measurement data storages

	programs - evaluation of existing tools and architectures	- formalization of the software measurement process level - survey about exiting measurement tools - survey among tool producers regarding measurement tool capabilities - the role of Software Measurement Databases (SMDB) in this regard

Table 3: Step 3 - Development and Operation

Preparation	**Execution**	**Analysis**
Evaluation of existing measurement paradigms Design of an architecture framework for dynamic assembled measurement infrastructures	- Study sources for automatic service composition - Enhancement of existing approaches to a holistic quality driven procedure - Use a stepwise mapping of ISO/IEC 15939 measurement process standard to different components - Creation of semantic knowledge for automated procedures - Integration of existing measurement databases for empirical analysis - Design of a measurement cockpit for empirical analysis	- description of the framework for the proposed infrastructure with different modeling languages (UML, BPMN) - requirements for measurement services - quality attributes for services - introduction of quality oriented service selection

	- Selection of a SOA-paradigm for measurement process automation	
Model verification	- Implementation of selected components for a proof of concept - Evaluation of measurement process level	- Framework related papers appear at international conferences

Table 4: Step 4 - Evaluation

Evaluation context	**Continuation of results**	**Future work**
Model validation	- Delphi study about the framework for a service-oriented measurement infrastructures	- Case studies of the usage of implemented components and empirical analysis about the usage of the framework in comparison to monolithic tools
Framework distributed along software tool manufactures, software measurement users and researchers	- prototypical implementation was integrated into software development environment via a plug-in approach and as services to demonstrate the concept of ubiquitous software measurement by the use of a service-oriented measurement infrastructure	- publication of the framework - integration of existing measurement tools to enhance provided functionality - providing semantic knowledge about software measures for support and automated procedures - proposal of a measurement SOA guideline for future measurement tool developments

1.4. Thesis Structure

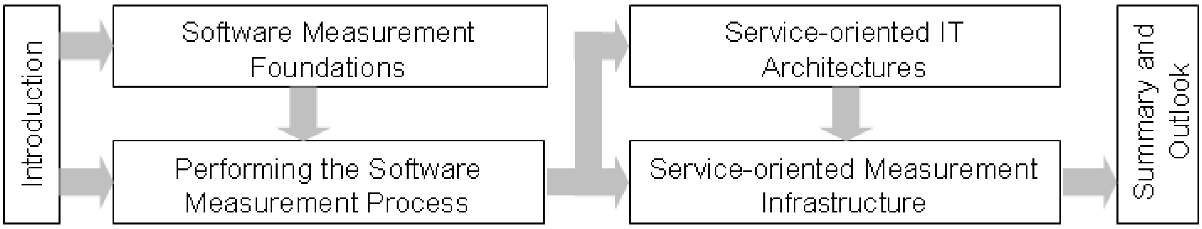

Introduction
Software Measurement Foundations
Performing the Software Measurement Process
Service-oriented IT Architectures
Service-oriented Measurement Infrastructure
Summary and Outlook

Figure 2: Thesis structure

The thesis in hand is divided into six parts. The structure of the thesis and the information path is shown in Figure 2. While the first chapter deals with motivation, scope, and goal of the thesis the subsequent chapters addresses the research work and is structured as follows:

Chapter two examines the foundation about software measurement, describes software measurement systems and processes, and analyzes their application in software measurement programs.

The third chapter reviews the current situation and state of the art regarding software measurement tools, measurement data repositories, and analyzes different approaches for software measurement.

Chapter four provides an overview about service-oriented architectures, examines the technology aspects, and reviews the SOA-capabilities of existing measurement tools.

The fifth chapter presents a framework for a Service-oriented Measurement Infrastructure including process description, an approach for a quality driven design of service-oriented architectures, and the presentation of a service-oriented measurement database.

Chapter six contains a summary of the research’ contribution, a review of the questions asked, and an outlook for future work.

2. Software Measurement Foundations

In particular experimental sciences bases on the possibility of quantifying certain aspects of theories to justify or reject assumptions. Centuries ago, Galileo Galilee (1564 - 1642) shaped the slogan:

> *"Measure what is measurable, and what is not measurable, make it measurable."*

Similarly for the software engineering discipline measurement more precisely software measurement has been identified as a key success factor for software engineering management. Tom DeMarco gives good evidence of its importance with his paraphrase "You can't control what you can't measure." [DeMarco82]

2.1. Metrics and Measures

In colloquial language measurement is a way of assigning a number, representing an attribute, to a physical object. In mathematics such assignment is usually called a mapping or a function.

For a lot of objects and measures the relationship between both is clear. If one tree trunk is five meters long and another one ten meters then the second one is longer. We can select the longer one even without doing measurement at all because of an intuitive perception of "length of a tree trunk".

But in most cases related to software engineering the relationship is anything but clear. For example we have to evaluate software systems consisting of different classes. One system contains ten classes each with complexity measures in the range of 30 to 40 and the other one 20 classes each with complexity measures in the range of 10 to 20. Which software systems contain less overall complexity?

With this example one can visualize the difficultness of an important engineering question: selection of different possible solutions. With appropriate measures software measurement can be very helpful to answer this type of questions. Usually there is more than one measure which determines the alternatives and tradeoffs between them have to be made.

The first example presented an intuitive perception regarding a specific measure but the second one showed that most cases in software measurement are not intuitive. The missing human ability of intuitive conclusions in every case is described in literature as an intelligence barrier.

Jürgen Kriz provided the following figure which illustrates the measurement process as the way to pass this barrier.

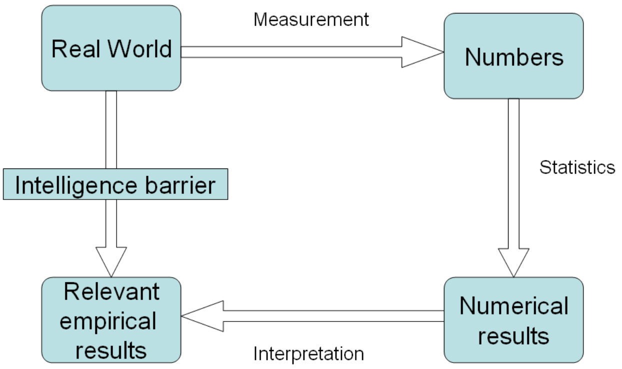


Figure 3: Measurement to pass the intelligence barrier [Kriz88]

The first step in this regard is to measure real-world objects to achieve numbers. The next step is to apply mathematical or statistical operations on the achieved numbers. The result is a second set of numbers which relates to the original object in the real world and the wanted conclusions. The last step is the interpretation of numerical result to achieve an answer in the objects domain.

Gary Ford gives a good example of this relationship which can be mapped to our previous one. Suppose you were given a dozen tree trunks with different lengths and asked to select the tree trunk of “average” length. Most likely it is not intuitively obvious which tree trunk one has to pick. If one measures the length of all tree trunks the result is a set of numbers. So after that one can compute the average (arithmetic mean) of those numbers and finally interpret this number as the length of the tree trunk to be chosen, and pick the trunk closest to this length. [Ford93]

For formal examinations the real-world objects with the included set of operations and relations is named as empirical relational system **A** with the following definition according to [Zuse1998]:

Let A be a non-empty set of objects, $\bullet\geq$ is an empirical relation on A and $\circ$ is a closed binary operation on A, the empirical relational system is defined as **A** $=(A,\bullet\geq,\circ)$**.**

The set of numerical objects with the included set of operations and relations is named numerical relational system **B**. With **B** $=(\Re,\geq,\otimes)$ where $\Re$ are the real numbers, $\geq$ a relation on A, and $\otimes$ stands for an arbitrary closed binary operation on $\Re$.

The process of measurement is defined as the one-to-one mapping μ from objects out of **A** to elements of **B**:

$$\mu : A \to \Re$$

with $$a \;\bullet\geq\; b \;\Leftrightarrow\; \mu(a) \geq \mu(b) \;\forall\; a,b \in A$$

The one-to-one mapping ensures that every object has exactly one measure.

In this case the tuple (**A, B,** μ) is called a scale. But there are a number of possible numerical representations for a given empirical relational system. If two numerical representations are acceptable measures the mapping from one measure to another is called admissible transformation [FentonPfleeger97].The set of admissible transformations that exist in each class of scale determines a scale type [Whitmire97].

In measurement five scale types can be named: nominal, ordinal, interval, ratio and absolute scale. In theory there are other scale types known but the mentioned four are sufficient for meaningful statistical operations [Zuse98]. Shari Pfleeger pictures the importance of determine the scale type with the phrase: *"Unless we are aware of the scale types we use, we are likely to misuse the data we collect."*[Pfleeger97].

Nominal scale type is the most primitive form of measurement and simply gives numeric terms to objects and any distinct numbering of elements is an acceptable measure but there is no notion of magnitude possible. Nominal-scale measures create a classification of elements but the classes are not ordered even if they are numbered from 1 to n.

In this regard the only informative operation for nominal-scale measures is the empirical equivalence relation $\approx$ for two objects $a, b \in A$ which is expressed by the numerical equivalence relation $=$. The formal expression of the nominal scale is:

$$((A, \approx), (\Re, =), \mu)$$

The set of admissible transformations spans over all one-to-one mappings that assign the same equivalence class to equivalent instances (with g is the set of admissible transformations: g = any one-to-one) [Zuse98].

Ordinal scale type measures assign numbers to objects in a specific order.

In addition to the empirical equivalence relation $\approx$ statements about the empirical ranking $\bullet \geq$ of two objects $a, b \in A$ are possible. $((A, \bullet \geq), (\Re, \geq, \mu)$

Ordinal scale $((A, \bullet \geq), (\Re, \geq, \mu)$

where for all $a, b, c \in A$

$a \bullet \geq b$ and $b \bullet \geq c$ => $a \bullet \geq c$ (transitive)

$a \bullet \geq b$ or $b \bullet \geq a$ (connected)

In the case of a transitive and connected relation $\bullet \geq$, $(A, \bullet \geq)$ is called a *weak order*. [Zuse98]

Because only ranking can be represented by ordinal scale type measures addition, subtraction and other arithmetic operations have no meaning.

Any mapping that preserves the order of the elements can be accepted. So, two measures can be related by a monotonic mapping and the class of admissible transformations is the set of all strictly monotonic mappings (g: strictly increasing monotonic function) [FentonPfleeger97].

Interval scale type measures enhancing the idea from ordinal scale in a way that the interval size between two empirical objects is meaningful throughout the range of values. In this case the gap between one class and another is reflected. For formal examination the term *algebraic structure* should be defined at first.

Let A be a non-empty set of objects and let $\bullet \geq$ be a quaternary relation on A (quaternary means for example a binary relation (AxA) on then the pair $(AxA, \bullet \geq)$ is called *algebraic structure* if for all $a, b, c, d, a', b', c', d' \in A$ the following axioms are sufficient satisfied.

- $(AxA, \bullet \geq)$ is a weak order.
- $ab \bullet \geq cb$ then $dc \bullet \geq ba$
- $ab \bullet \geq a'b'$ and $bc \bullet \geq b'c'$ then $ac \bullet \geq a'c'$.
- $ab \bullet \geq cd \bullet \geq aa$, then there exists $d', d'' \in A$, such that $ad' \approx cd \approx d''b$.
- $a_1, a_2 ... a_i, ...$ is a strictly bounded sequence ($a_{i+1}, a_i \approx a_2 a_1$ for every a_i, a_{i+1} in the sequence; not $a_2 a_1 \approx a_1 a_1$; and there exist $d', d'' \in A$ such that $d'd'' \bullet \geq a_i a_1 \bullet \geq d''d'$ for all a_i in the sequence), then it is finite.

To complete the formal view the interval scale can be defined as ordered containing the algebraic structure $(AxA, \bullet \geq)$ the numeric relational system **B** and the measure μ.

Interval scale $((AxA, \bullet \geq), (\Re x \Re, \geq), \mu)$

As an enhancement in comparison to ordinal scale type addition and subtraction are acceptable and according to Zuse any positive linear function (*g(x)=ax+b, a>0*) is an admissible transformation [Zuse98].

Ratio scale type measures assign values in a way that preserves the ratio of scale values. In doing so, ratio scale type is highest scale type of measurement. Ratio scale type is common in other academic fields like physics. For example “length” is a ratio scale type no matter of the measurement unit because ratio concepts are meaningful (e.g. “twice as long”).

The main characteristic is a zero element which stands for the total lack of the measured characteristic. This zero element is a consequence of the introduction of an additive property over a concatenation operation which increases over equal intervals.

For formal examinations it is important to define the term *closed extensive structure* [Zuse98]:

Let A be a non-empty set of objects and let $\bullet \geq$ be a binary relation on A, and let $\circ$ be a closed binary operation on A then the ordered triple $(A, \bullet \geq, \circ)$ is called a *closed extensive structure* if there is a function μ on A on the domain $\Re$ such that for all $a, b \in A$:

- $a \bullet \geq b \Leftrightarrow \mu(a) \geq \mu(b)$
- $\mu(a \circ b) = \mu(a) + \mu(b)$

Additionally the function μ' satisfies both statements, when there exists $\alpha > 0$ such that $\mu'(a) = \alpha\mu(a)$.

The formal expression of the ratio scale then is: $((A, \bullet \geq, \circ), (\Re, \geq, \otimes), \mu)$ where the described axioms of the extensive structure are valid.

The above described function μ' defines the only admissible transformation for ratio scale measures which can alternatively be defined as positive linear functions in the form (*g(x)=ax, x>a*).

The last considered scale type is the **absolute scale**. It is the most restrictive scale type of all mentioned. The only way of measuring objects in this case is the counting of the number of elements and the only admissible transformation is the identity transformation (*g(x)=x*) [Zuse98].

For formal expression let **A** $= (A, \bullet \geq, o_1, o_2, ...)$ be a totally ordered structure than μ is an absolute scale type measure for **A,** if there exists a numerical relational system **B** that is the set of N-representations for **A** that are onto **B**. The numerical relational system **B** has exactly one element.

The described sequence of scales is increasingly restrictive. Every interval scale is also an ordinal scale, but not vice versa. Every ordinal scale is also a nominal scale, but not vice versa. A summary of the presented scale types with the defined relations, applicable statistics and admissible transformations is shown in Table 5.

Table 5: Measurement scale types and admissible transformation

Defined Relation	**Type of Scale**	**Applicable Statistics**	**Admissible Transformation**
Equivalence	Nominal	Frequency distribution, contingency	Any one-to-one
Equivalence, greater than or equal	Ordinal	Distribution, Median, Kendall and Spearman Correlation, association	g: strictly increasing monotonic function

Equivalence, greater than or equal ratio value within an interval	Interval	Arithmetical median, standard deviation, Pearson correlation, multiple correlation	g(x)=ax+b, a>0
Identity, greater than or equal, ratio value as per two values	Ratio	Geometrical median, coefficient of variation, all of the above	g(x)=ax, a>o
Identity	Absolute	Number of occurrences	g(x)=x

Having introduced measures and scale type another term often linked with measurement is metric. A metric measure the distance between two points and has an accurate mathematic definition: A function *f* is called metric if it satisfies the following three properties, where (x,y) is the distance between two points x and y:

f(x,y) = 0 if x=y (equity)

f(x,y) = f(y,x) for all x,y (symetric property)

f(x,z) <= f(x,y) + f(y,z) for all x,y,z (triangular inequation)

A familiar example for a metric is the metric of Euclid which measures the shortest distance between two points on a plane:

Let $p_1 = (x_1, y_1)$ and $p_2 = (x_2, y_2)$ be two points in a plane. Then the distance d is defined by Euclid as: $d(p_1, p_{2)} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$.

A 2nd example is the so called "Manhattan" metric which calculates the distance between two points that one would travel on a grid layout of streets (for example on idealized Manhattan Island, New York).

With the above defined two points on a plane the metric is defined as: $m(p_1, p_2) = |x_1 - x_2| + |y_1 - y_2|$.

The following figure exemplifies both metrics. The Manhattan metric as the way a car or a person has to drive or walk to reach a specific location in a checker-board pattern built town like Manhattan and the Euclid distance as the air-line distance between the same points.

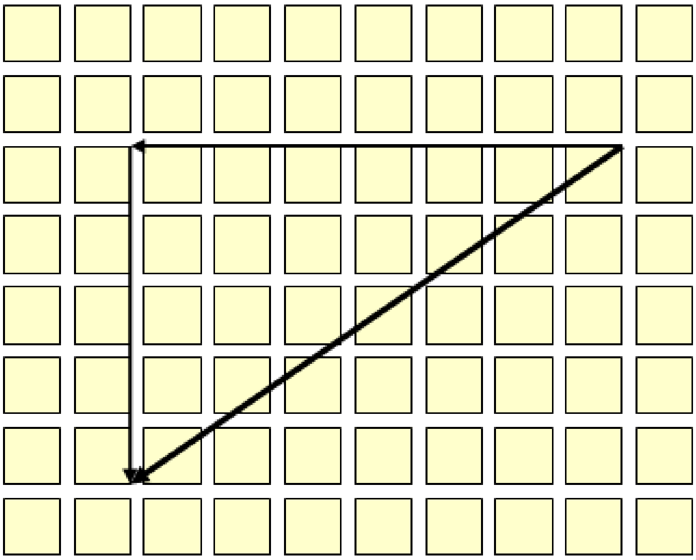


Figure 4: Difference between Euclid distance and Manhattan metric

Some but not all software measures are metrics and because of that in software engineering the term is ambiguous. Some researchers decline the usage of the term "metric" because "The use of the term "metric" for any other type of measure is imprecise at best and misleading at worst." [Whitmire97] They argue that the term "metric" should be reserved for the area of geometry in mathematics. Other researchers refuse ignoring the term and tried to find a solution by adapting the meaning. Definitions like "metric is a measurement function" [KanerBond04] or "software metric is a term that embraces many activities, all of which involve some degree of software measurement" [FentonPfleeger97] imply that a software metric is an application of some kind of software measurement.

Summary: Metrics and Measures

Out of the presented viewpoint Software Metrics can be defined as: *"The continuous application of measurement-based techniques to the software development process and its products to supply meaningful and timely management information, together with the use of those techniques to improve that process and its products."* [Goodman04]

But since the difference of metrics and measures is not essentially [Zuse98] both terms "software metric" and "software measure" will be used as synonyms in the thesis in hand.

2.2. Measurement Sources

Having introduced the mathematical foundations of software measures the mapping to the practical implementation in software engineering is presented next. The term software measurement is defined as: *"Software measurement is the process of the quantification of* *software engineering attributes according to selected measurement goals by using (if possible) appropriate tools."* [Dumke04]; [FentonPfleeger97], [AbranMoore$^+$04]

After introducing software management as the application of management activities (among others namely measurement) the classification of the software management areas and the interconnections are shown in Figure 5.

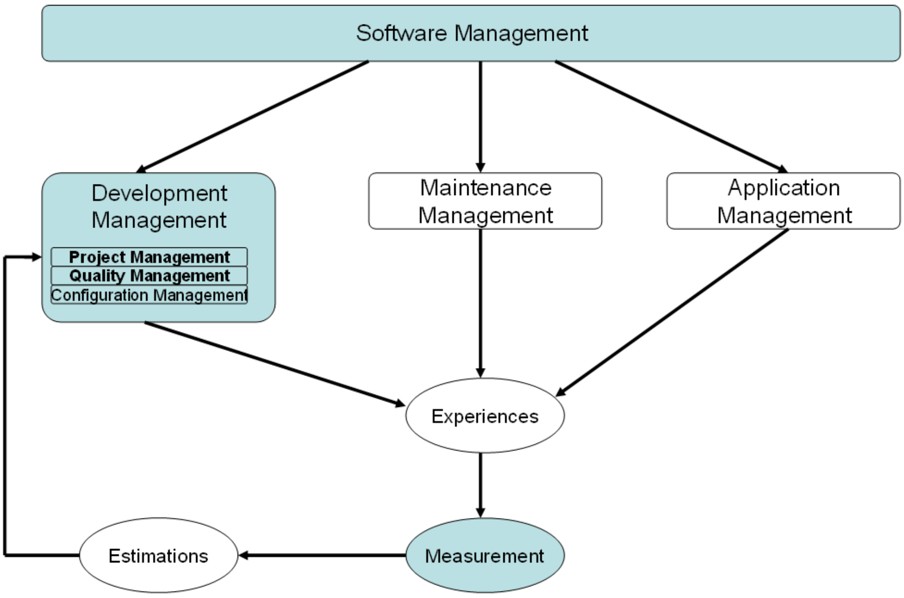


Figure 5: Software management areas [Dumke04]

Out of this classification the main areas for the comprehension of software measurement are project management and quality management.

Software project management thereby contains *the planning, observation and controlling of software projects for the supply of resources on schedule and the realization of a software product on time and budget.* [Dumke04], [Wallmüller01]

Software quality management means *the assurance of quality characteristics for a software product on basis of process and resource quality by the application of organizational methods, chosen techniques and technologies.* [Dumke04] [Wallmüller01]

Whereas software quality different meanings and can be defined in various ways [Dumke03], [DeMarc89], [Weinberg92].

For the thesis in hand the definition from ISO has been used where software quality is defined as *the totality of features and characteristics of a product or service that bear on its ability to meet stated or implied needs [ISO/IEC94].*

Thus, the quantification of instances of entities which are involved in some way in a software engineering process is the intention of software measurement. The question what such entity can be and what characteristics can be measured was of substantial interest for the software measurement community over years.

Norman Fenton presented in 1991 a framework introducing three classes for software engineering entities namely *products*, *processes* and *resources [Fenton91].* Some researchers enhanced the framework by adding *projects* or exchanging *resources* with *projects* [Whitmire97] [Daskalantonakis94].

Since the framework from Norman Fenton is still the most accepted and feasible one, it is used for the thesis in hand.

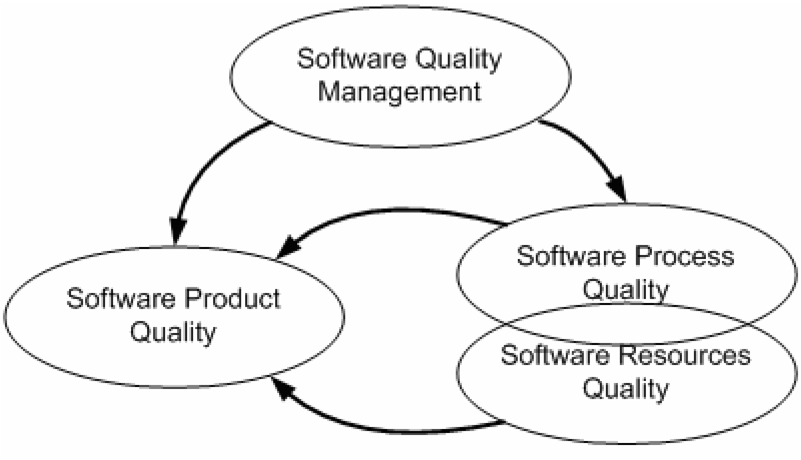


Figure 6: Interdependences between entities [Lother07]

2.2.1. Product Measurement

The measurement of software engineering products can be seen as the origin of software measurement. Information about final product characteristics satisfies the costumers desire to know as much as possible about used commercial software [Pfleeger97]. Examples for such measures are the number of implemented lines of source code (LOC), the metrics from Halstead (volume, effort), or the cyclomatic complexity by McCabe [McCabe76]. In the course of time fewer and fewer people wanted to wait until final products were available and many software engineering projects failed to deliver some kind of a final product. Beside simply applying the above described metrics to semi-products, measures has been evolved to evaluate early stages of software engineering.

Considering the early stages of software engineering quantitative information about the quality of requirement specifications defined by Mora and Denger [MoraDenger03] can provide useful information.

Afterwards, quantitative information about the emerged design of software has been identified as important. Measures for design complexity (Henry and Kafura´s structure metrics) [HenryKafura81] or several object-oriented measures for design quality (Chidamber and Kemerer´s metrics suite for object-oriented design) [ChiKem94] have been invented to answer this need.

The ISO/IEC 9126 international standard for product quality and the successor ISO/IEC 25000 provide a comprehensive compendium about software product quality attributes and measures [ISO/IEC01]. The ISO/IEC 9126 standard is divided into four parts.

Part1: Software Engineering Product Quality - Quality model: describes the relationship between different approaches to quality and describes the quality characteristics and sub-characteristics of a distinct software product [ISO/IEC01].

Part2: Software Engineering Product Quality - External metrics: describes the external metrics used to measure the characteristics defined in Part 1 [ISO/IEC03a].

Part3: Software Engineering Product quality - Internal metrics: describes the internal metrics used to measure the characteristics defined in Part 1 [ISO/IEC03b].

Part4: Software Engineering Product quality - Quality in use metrics: identifies the measures used to distinguish the effects of the quality characteristics for the user [ISO/IEC04].

Table 6: ISO/IEC 9126 Internal and external quality characteristics

Functionality	**Maintainability**
• Suitability • Accuracy • Interoperability • Security • Functional compliance	• Analyzability • Changeability • Stability • Testability • Maintainability compliance
Efficiency	**Reliability**
• Time behavior • Resource utilization • Efficiency compliance	• Maturity • Recoverability • Fault tolerance • Reliability compliance
Usability	**Portability**
• Understandability • Learn ability • Operability • Attractiveness • Usability compliance	• Adaptability • Install ability • Co-existence • Replace ability • Portability compliance

The ISO/IEC 9126 does not treat the product isolated and the connections to other entities are shown in Figure 7.

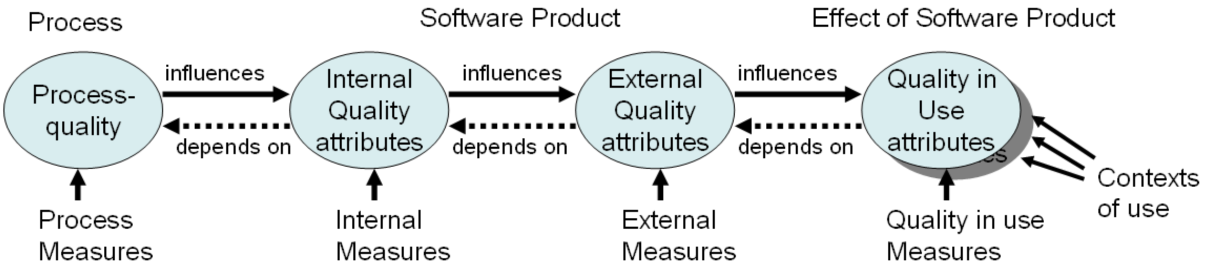


Figure 7: ISO/IEC 9126 Quality in product life cycle [ISO/IEC01]

Whereas the internal quality attributes measure a product by means of the principles of static analyzes. The static analysis contains syntactic and semantic analysis of the source code and delivers results for complexity measures, call graphs, control graphs, quality reports, and so on. The principle of static analysis is shown in Figure 8.

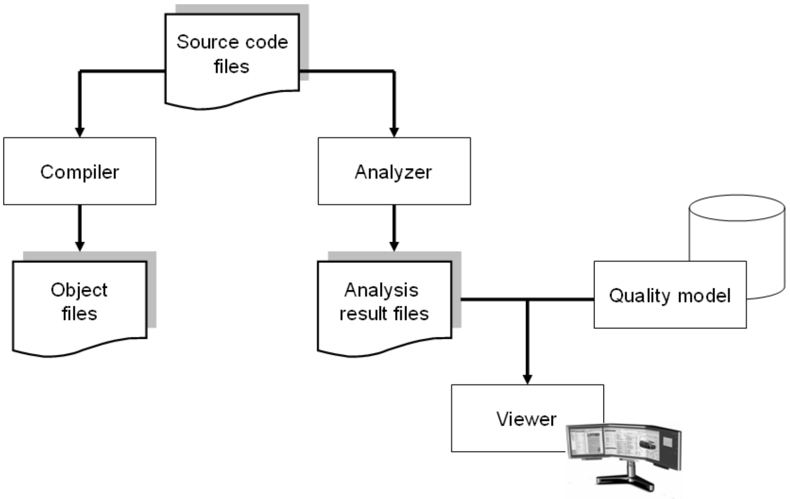


Figure 8: Principle of static source code analysis [EbertDumke07]

In contrast to this procedure dynamic source code analysis measures the running program. Previous the compiling the source code statements are marked and instrumented. During the execution information are written into the execution result file. Afterwards information about test coverage and other dynamic measures can be analyzed. This dynamic source code analysis measures external quality attributes and therefore all ISO/IEC 9126 external quality characteristics are applicable. The principle of dynamic source code analysis is shown in Figure 9.

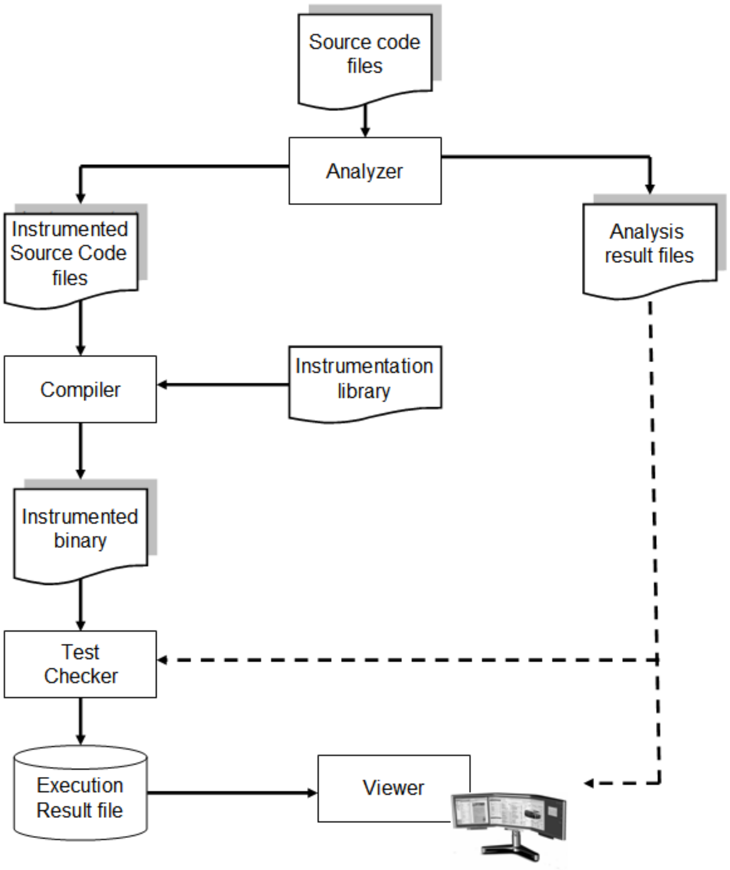
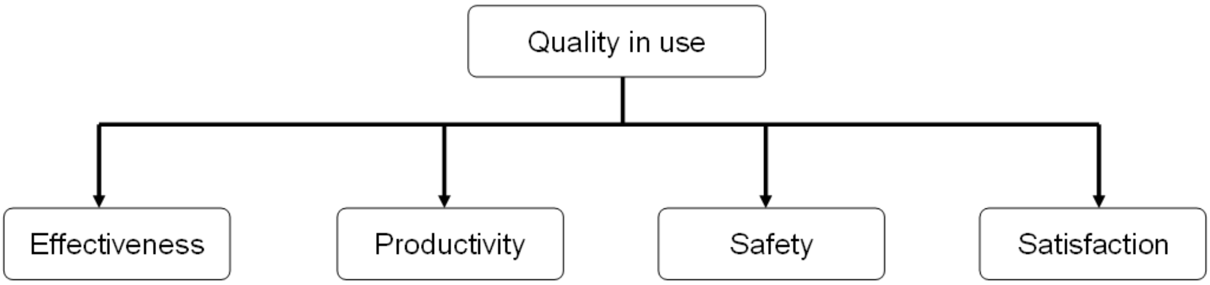


Figure 9: Principle of dynamic source code analysis [EbertDumke07]

Additional to external and internal quality attributes the product quality can be determined by measuring the effect of a software product in its context of use. The characteristics taken into account for the quality in use model are shown in Figure **10**.



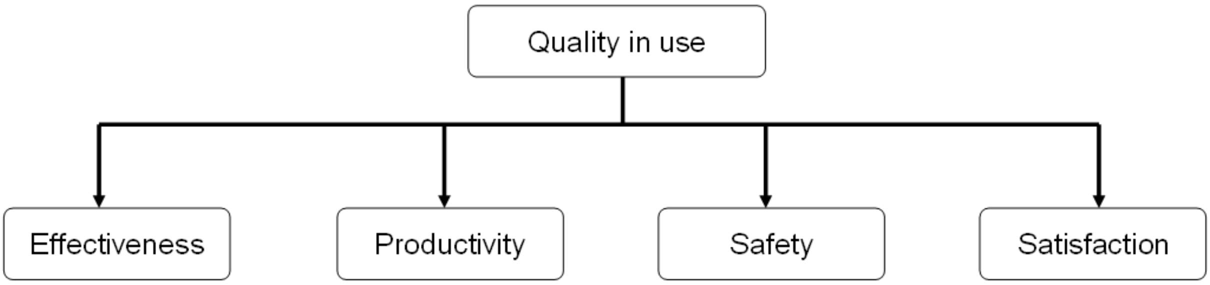


Figure 10: ISO/IEC 9126 Quality in use characteristics [ISO/IEC04]

Due to the great spread of different stakeholders, users, and use-cases of software products, quality can mean something completely different for different kinds of software. Some attributes are of higher or lower relevance and it is not the silver bullet to measure all of ISO/IEC 9126 over 300 measures each time when quality evaluation is needed.

To tackle this problem the definition of so-called quality models arises as early as product measures have been established. The usual procedure thereby is to specify quality requirements and to make trade-offs between software product capabilities [ISO/IEC01] [BoehmBrown+76] [McCallRichards+77].

The result is a quality model containing a set of measures with aligned thresholds for different quality characteristics and possible different weighting of these characteristics. [BuglioneAbran99]

As an example for a set of measures for a distinct area of product characteristic the metric suite for object-oriented design should be presented next.

This metrics suite was proposed in [ChiKem94] by S. R. Chidamber and C. F. Kemerer. The six structural design metrics proposed by them are:

- Weighted Method per Class *(WMC)*

  WMC is the sum of complexities of all methods in a class. Consider a class *C1* with methods *M1, . . . ,Mn* that are defined in the class. Let *c1, . . . ,cn* be the complexity of each of these methods. Then WMC has been defined as:

  $$WMC = \sum_{i=1}^{n} c_i$$

- Depth of Inheritance Tree *(DIT)*

  The *DIT* of a class is defined as the longest way from a class in the inheritance tree to the node.

- Number of Children *(NOC)*

  The Number of immediate sub-classes subordinated to a class in class hierarchy.

- Coupling between Objects *(CBO)*

  CBO for a class is the count of the number of other classes to which it is coupled. Two classes are coupled together if methods of one use methods or instance variables of another. Excessive coupling between object classes is negative to modular design and inhibits the reuse of the class in other applications. As more independent a class is, as higher is the reusability.

- Response for a Class *(RFC)*

  $RFC = |RS|$ where RS is response set for the class. This is a set of methods that can potentially be executed in response to a received message. This message is sent by an object of that class. Additionally calls from outside the class can be included. Thus, RFC is also a measure for the communication between on class and another.

- Lack of Cohesion in Methods *(LCOM)*

  The *LCOM* is the count of the number of method pairs whose similarity is *0* minus the count of method pairs whose similarity is not zero. The degree of similarity for two methods M_1 and M_2 in class C_1 is defined by:

  $\partial() = \{I_1\} \cap \{I_2\}$ where $\{I_1\}$ and $\{I_2\}$ are the set if instance variables used by M_1 and $M_{2.}$

  As larger the number of similar methods is as more cohesion exists between these two classes. In object-oriented design the cohesion should be as high as possible.

The interplay between coupling and cohesion is shown in Figure 11. The right hand side represents the object-oriented ideal.

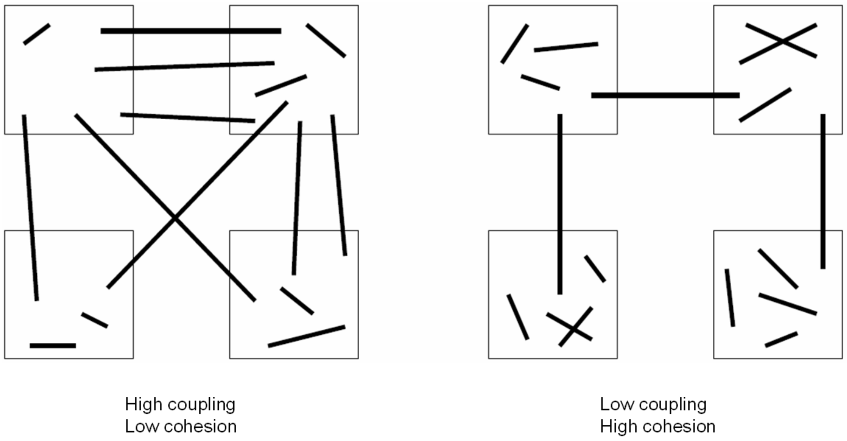


Figure 11: Interplay between coupling and cohesion [ChiKem94]

2.2.2. Resource Measurement

Beside the product itself, the used resources to create the products are of high importance. The primary domain is thereby human resources: skills or abilities as well as difference in mental conditions of involved people results in variation of quality and complexity of created products [Basili1996], [DeMarco89].

Alongside with human factors non-human factors like time schedule, production equipment (development tools, working environment) and monetary budget are important factors, too [ZenkerKunz+08a].The following Figure 12 shows a possible distribution of the different characteristics addressed to the main parts of the software development resources.

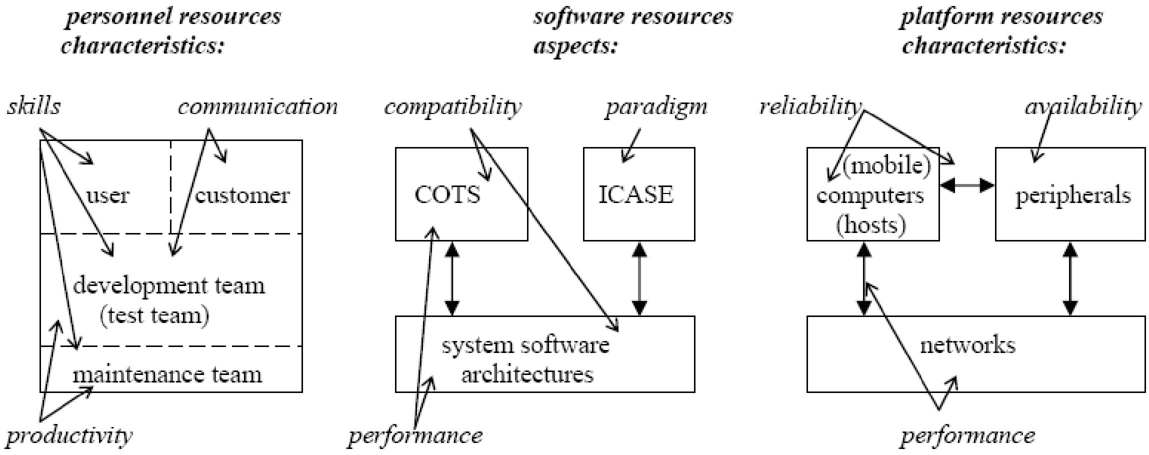

personnel resources characteristics:
software resources aspects:
platform resources characteristics:
skills
communication
user
customer
development team (test team)
maintenance team
productivity
compatibility
paradigm
COTS
ICASE
system software architectures
performance
reliability
availability
(mobile) computers (hosts)
peripherals
networks
performance

Figure 12: Basic visualization of resource characteristics [DumkeKunz+08]

To control and predict the software development measurements of resources quantifications as well as prediction of future trends are of substantial interest [ZenkerKunz+07]. Measurements of personal resources can be done according to the Personal Software Process (PSP) where the activities of stakeholders are recorded in an auto didactical process [Humphrey96].

For cost and schedule related aspects there exists a complete research area called effort estimation. Models like the Constructive Cost Model (COCOMO) with different variations (COCOMO 2.0 and COCOMO II) [Boehm00] or the Functional Size Measurement Model with different methods (e.g. IFPUG Function Points [IFPUG99], COSMIC Full Function Points [AbranOligny+00]) provide techniques for measuring and estimation of financial costs, project duration, or head count [Kunz+07b].

2.2.3 Process Measurement

In the course of time deeper insights of researchers' leads to the meaningfulness of taking into account not only resources and products but also the activities which lead to these products [Wallmüller01].

Considering theses activities it is important because of "product quality is evidence of process success" [Pfleeger97].

But the definition of "process" is not beyond controversy in software engineering research community. As mentioned in the entity classification some researchers prefer the term "project" or embrace a time factor. To avoid misunderstanding a well-defined demarcation is needed.

According to [Pall87] [Lonchamp93] [AbranMoore+04] a process can be defined as the logical organization of people, materials, energy, equipment, and procedures into work activities designed to produce a specific end result.

In contrast to that definition projects bear references to the time and are nonrecurring while processes are based on repeatable activities. They can be defined as a distinct execution of a software development process with a set amount of resources within a set interval [Whitmire1997]. That means that a project is a set of entities and not an entity given alone.

Summary: Product, Resource, and Process Measurement

Embracing the aspects the software product (SP) can be defined as:

$$\boldsymbol{SP} = (M_{SP}, R_{SP}) = (\{programs, documentations\}, R_{SP})$$

Even though product measures are these days able to quantify a lot of different aspects, one has to take into account other entities to get a complete picture.

For formal considerations the software development resources *SR* can be defined as:

$$\boldsymbol{SR} = (M_{SR}, R_{SR}) = (\{personnelResources, softwareResources, platformResources, financialRessources\}, R_{SR})$$

whereas the software resources play a dual role in the software development: as a part of the final system (as COTS or software components) and as the support for the development (as CASE or integrated CASE as ICASE).

Dumke et al. [DumkeKunz$^+$08] defines the software development process (SD) as the collaboration of elements from development method, lifecycle model, software management, and needed resource entities (SR).

$$\boldsymbol{SD} = (M_{SD}, R_{SD}) = (\{developmentMethods, lifecycle, softwareManagement\} \cup M_{SR}, R_{SD}).$$

A formal notation of a software development project (SDP) can be defined according to [DumkeKunz08] as following:

$$\boldsymbol{SDP} = ((M_{SDP}, R_{SDP}) = (M_{SP} \cup M_{SR} \cup M_{SD} , R_{SDP})$$

Measureable attributes in this regard are for example number of resource usage, mean cost per detected error, number of coding faults found. But the boundaries between process and project measures often overlap since some of the measures contains as time reference and thereby quantifying a process by measuring its application in concrete projects.

2.3. Software Measurement Systems and Processes

The formal representation of the synthesis of the above elements can be defined as a Software Engineering Measurement System (SEMS) [DumkeKunz08].

$$\boldsymbol{SEMS} = (M_{SEMS}, R_{SEMS}) = (\{G, A, M, Q, V, U, E, T, P\}, R_{SEMS})$$

where *G* is the set of the measurement goals, *A* the set of measured artifacts or measurement objects, *M* the set of measurement methods, objects or entities, *Q* the set of measurement quantities, *V* the set of measurement values (especially we could have the situation *Q* =*V*), *U* the set of measurement units, *E* the set of measurement-based experience, *T* the set of measurement CASE tools (respectively CAME tools), and *P* the set of the measurement personnel. R_{SEMS} defines all meaningful relations between the elements of M_{SEMS}.

Especially, the *measurement process MP* as one of the instantiations of a software measurement system is explained by the following sequence of relations

$\boldsymbol{MP}$: $(G \times A \times M)_{T,P} \rightarrow (Q \times E)_{T,P} \rightarrow (V \times U)_{T,P} \rightarrow E' \times A'$

This measurement process description explains the process results as quantities including some thresholds, values involving their units and/or extended experiences combined with improved or controlled measurement artifacts.

Software measurement process is embedded in the general motivation and classification characterized in the following figure.

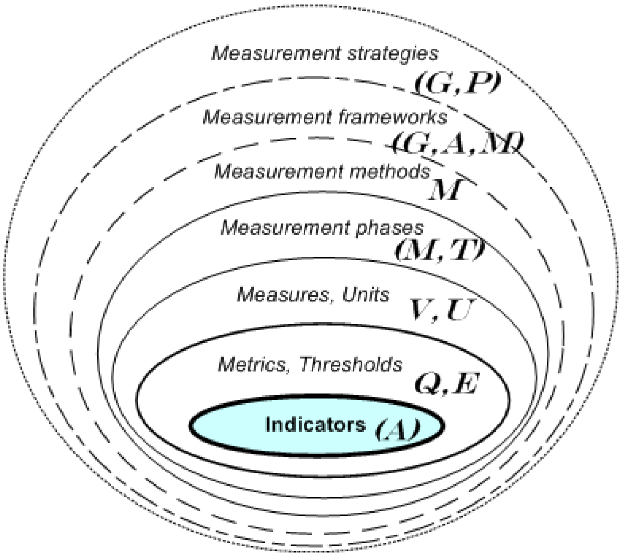


Figure 13: The general layer model of software measurement

Furthermore, the detailed phases of software measurement and their different kinds of measurement methods can be described as the following.

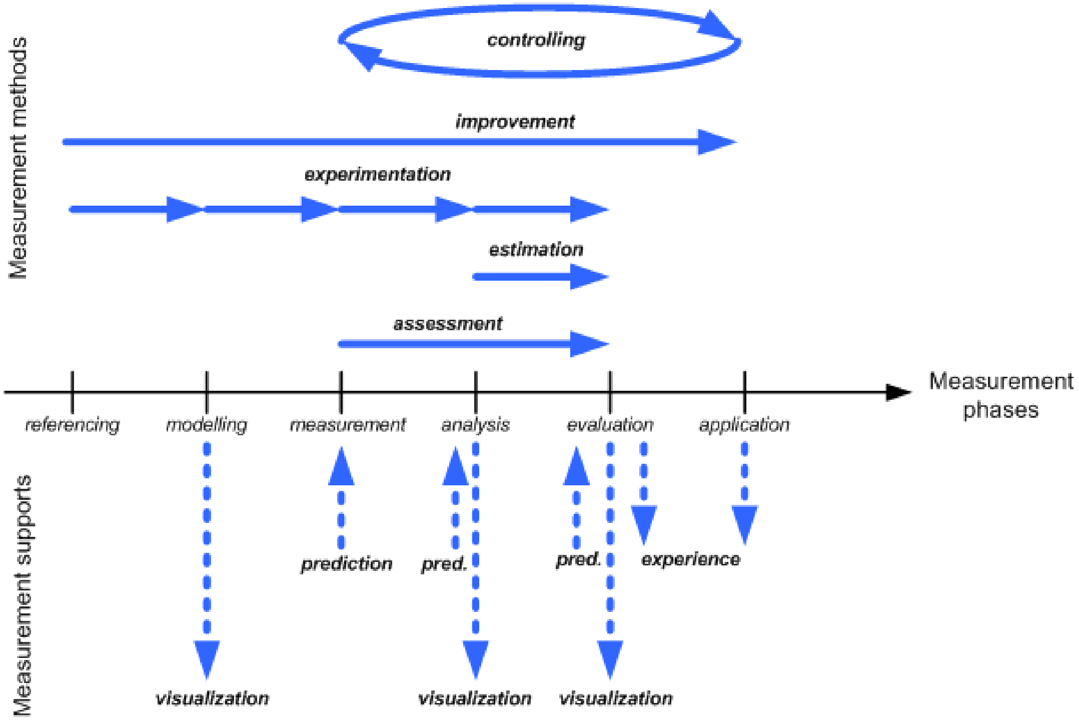


Figure 14: Software measurement phases and methods

2.3.1. Measurement Ingredients

The tuple of (***G*** × ***A*** × ***M***) describes the input and basis for any software measurement. The detailed characteristics of these three sets are [DumkeKunz08]:

G: ***Intention:*** Goals are considered as *understanding, evaluation, improving* and *managing.* This enumeration corresponds to an increasing level of measurement goals.

Viewpoint: On the other hand the goals depend on the special viewpoint such as *internal goals/quality, external goals/quality* and *goals/quality in use.* [ISO/IEC01]

A: ***Domain:*** The considered measurement artifacts should be the general classification of software as *products (systems), processes* (e. g. *project)* and *resources* (including their different parts or aspects (e. g. *product model, process phases* or *personal resources*)). [Fenton91]

Origin: Note that a pendant or analogical artifact of measurement is considered to drive the path to the kinds of measurement as *analogical conclusion*. Analogy can be defined as *tuning* (where a pendant in the same class of software systems is used) and as *adaptation* (where another pendant of artifact is used). This kind of description is motivated in the following consideration.

The complexity of the measured artifact is explained as: Software measurement of different systems is related to the kind of systems (information-based, embedded, web-based, decision support, knowledge-based etc.) and to the different kinds of software development paradigms such as object-oriented software engineering (OOSE), aspect-oriented programming (AOP), component-based software engineering (CBSE), feature-oriented development (FOD), service-oriented software engineering (SOSE), event-based design (EBD) and agent-oriented software engineering (AOSE).

In contrast, general characteristics of software systems are meaningful in different IT environments such as performance, security, and usability or context-dependent as outsourcing and off shoring. And finally, measurement artifacts can depend upon different kinds of systems such as embedded systems and information systems etc.

M: ***Method:*** The chosen measurement methods are classified here as *experiment/case study, assessment, improvement* and *controlling.* That means that measurement should contain the partial phases as *referencing, modeling, measurement, analysis, evaluation* and *application* and could cover different parts of these phases. Note that the dominant use of experiences could lead to the kinds of measurement as *estimation* or *simulation.*

Sort: Furthermore, depending on the measured artifact(s) that is involved in the measurement it will be distinguished between *no measurement* (no artifact), *aspect-oriented measurement* (considering some aspects of product or process or resources), *capability-oriented measurement* (considering the whole product, the whole process or all resources) and *whole measurement* (considering all, product and process and resources).

Measurement Output

The immediate output of software measurement consists of numbers that would be interpreted by using any experience described by the pair as ($Q \times E$). The typical properties of these sets are:

Q: ***Value:*** This set of metrics values/numbers characterizes a *qualitative measurement* and are given in a *nominal scale* or *ordinal scale.*

Structure: Measured values could be structured in different kinds of presentations and transformations such as *tuple, table, aggregation* and *normalization.*

E: ***Form:*** The appropriate experiences for *Q* are given as *analogies, axioms, correlations, intuitions, laws, trends, lemmas, formulas, principles, conjectures, hypothesis's* and *rules of thumb.*

Contents: The contents or kinds of experience could be *thresholds, lower and upper limits, gradients, calculus* and *proofs.*

Measurement Results

As a higher level of measurement output the goal is to achieve real measures including their units. Characteristics of the sets in the tuple ($V \times U$) are:

V: ***Measure:*** This set of metrics values characterizes a *quantitative measurement* and is given an *interval scale* or *ratio scale.*

Aggregation: The values could be built as different structures and aggregations such as *measurement repositories, simple visualizations* (e. g. diagrams scatter plots)*, dashboards* and *cockpits.*

U: ***Type:*** The measurement unit could be *CFP* (COSMIC FFP functional size), *program length* of Halstead [Halstead77], *kilo delivered lines of code* (KDSI)*, cyclomatic complexity* of McCabe [McCabe76] etc.

Standard: Otherwise the mostly used units could be classified as physical, economical, sociological, software and hardware units.

Measurement Resources

Every phase of the software measurement process is supported by tools used by personnel. The detailed characteristics of these sets are:

T: ***Level:*** The kind of tool and the tool support should be classified as *manual* (without any tools), *semi-automatic* and *automatic.*

Support: In contrast, the tool could be applied in the IT area (as *internal measurement*) or by vendors (as *external measurement*).

P: ***Kind:*** The measurement personnel involve the different kinds of measurement and intentions and could be distinguished as *measurement researchers, practitioners* and *managers.*

Area: Furthermore the measurement personnel is divided in origin measurement staff (measurement analyst, certifier, librarian, metrics creator, user and validator) and in IT staff who use the software measurement indirectly (administrator, analyst, auditor, designer, developer, programmer, reviewer, tester, maintainer, customer and user).

Measurement Repercussions

Finally, the software measurement is leading to extensions of the experience and to improvements of the measures artifacts explained in the tuple ($E' \times A'$). Typical properties are:

E': **Form:** The obtained experiences are also given as *analogies, axioms, correlations, intuitions, laws, trends, lemmas, formulas, principles, conjectures, hypothesis's* and *rules of thumb.*

Extension: Especially the marked set of experiences explains the extended knowledge based on the measurement, evaluation and exploration and can produce *formula correction, principle refinement, criteria approximation* and *axiom extension.*

A': ***Domain:*** The kinds of measurement that include the change or improvement of the measured artifacts leads to such a marked set *A.*

Changing: Depending on the measurement process goals and methods, the artifact could be *understood, evaluated, improved, managed* or *controlled.*

The measurement process *MP* itself should be characterized by the level of covered/measured artifacts (as ***approach***) and by the kind of IT relationship (as ***solution***). Hence, the essential measurement process characteristics are defined in the following formal manner [DumBra+07]:

$$\boldsymbol{MP} = (G^{\substack{intention\\viewpoint}} \times A^{\substack{domain\\origin}} \times M^{\substack{method\\sort}})_T{}^{\substack{level\\support}}{}_{,P}{}^{\substack{kind\\area}} \rightarrow (Q^{\substack{value\\structure}} \times E^{\substack{form\\contents}})_T{}^{\substack{level\\support}}{}_{,P}{}^{\substack{kind\\area}} \rightarrow$$
$$(V^{\substack{measure\\aggregation}} \times U^{\substack{type\\standard}})_T{}^{\substack{level\\support}}{}_{,P}{}^{\substack{kind\\area}} \rightarrow E^{\substack{form\\extension}} \times A^{\substack{domain\\changing}}$$

2.3.2. Measurement in Software Development Process

As mentioned above the quality of a software development process intensely influences the quality of the resulting product. That leads to several approaches for evaluating process quality [Humprey87]. Important standards and methods are the ISO/IEC 900x series (including [ISO/IEC00a] and [ISO/IEC00b]) and the CMMI (Capability Maturity Model Integration) [SEI02a] [SEI02b]. These models take into account the key role of the logical organization of all activities for the software development process and therefore integrates product, resource or project related sub processes into a holistic process model.

2.3.2.1. ISO/IEC 900x Series

ISO/IEC 900x is not limited to software development processes but used in different fields of application. ISO/IEC published a survey stating that nearly 800.000 organizations have been evaluated by ISO/IEC 9001:2000 [ISO/IEC05].

The general goal of the standard is to describe minimal characteristics of quality and process management to establish a quality system [Pandian2003]. Such quality system is often a basic business precondition for customers before ordering products or services.

For the development process especially the ISO/IEC 9000-3 gains importance because the standard connects the ISO/IEC 900x series to the software engineering to other IT-related standards. The different documents and their affiliation to the different series are presented in Figure 15.

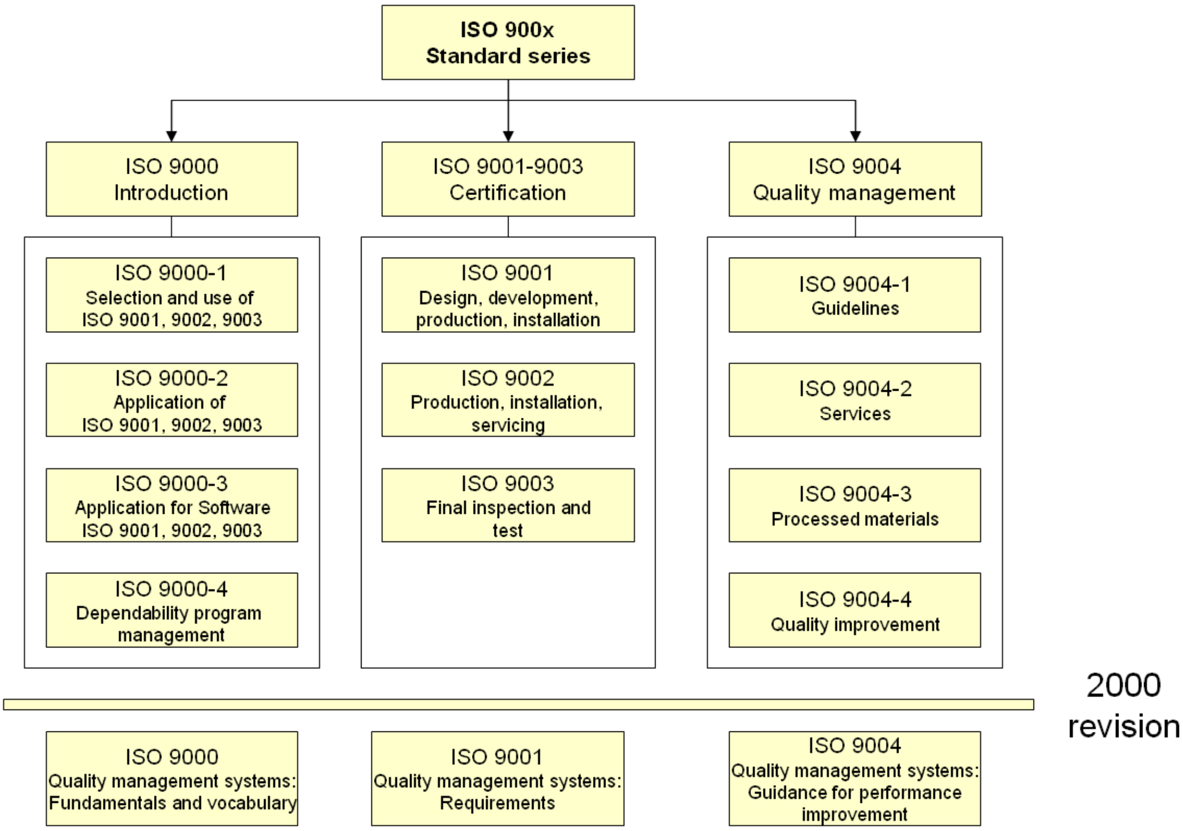


Figure 15: ISO 900x series

The ISO/IEC 900x series quality management system aims a continuous improvement to satisfy customer requirements. The basic activities of this process-oriented quality management system are shown in Figure 16.

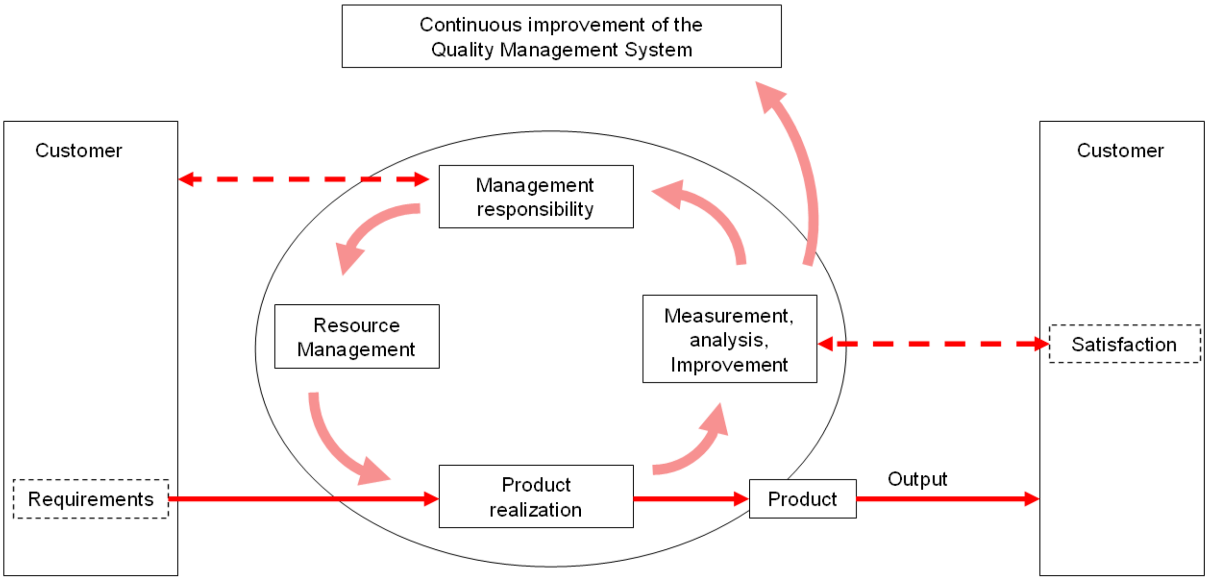


Figure 16: Process-oriented Quality Management System

2.3.2.2. CMMI Framework for Process Integration and Product Improvement

In contrast to the ISO/IEC 900x series the Capability Maturity Model Integration has been developed specific for the software development process. The model is originated from the Capability Maturity Model (CMM) which has been developed by the Software Engineering Institute at the Carnegie Mellon University for the U.S. Department of Defense to choose among different software development companies [Humprey87].

Alongside with the mentioned CMM, the Systems Engineering Capability Model (SECM), and the Integrated Product Development CMM (IPD-CMM) has been combined to the CMMI Framework. [SEI02a] [SEI02b]

Due to different approaches in the initial models two variable representations for the underlying process has been provided: staged and continuous [ChrissisKonrad+03].

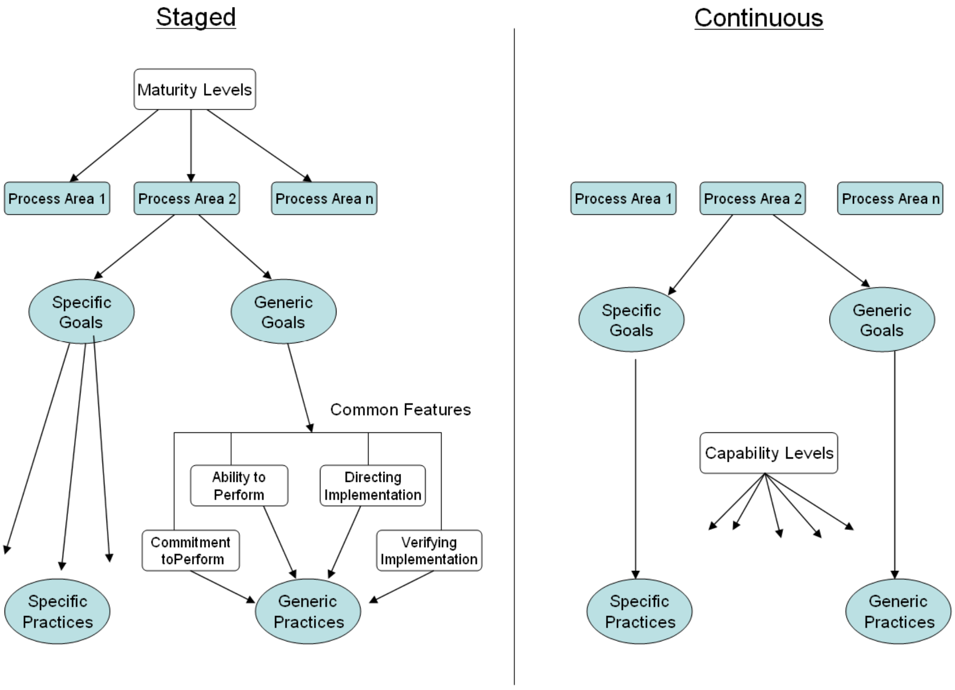


Figure 17: CMMI staged and continuous structure

The different order of the elements in the two structures is compared in Figure 17. The unique maturity levels of the staged representation, the focus of the different stages, and the aligned processes are shown in Figure 18. The process areas and the capability levels of the continuous representation are presented in Figure 19 and Figure 20.

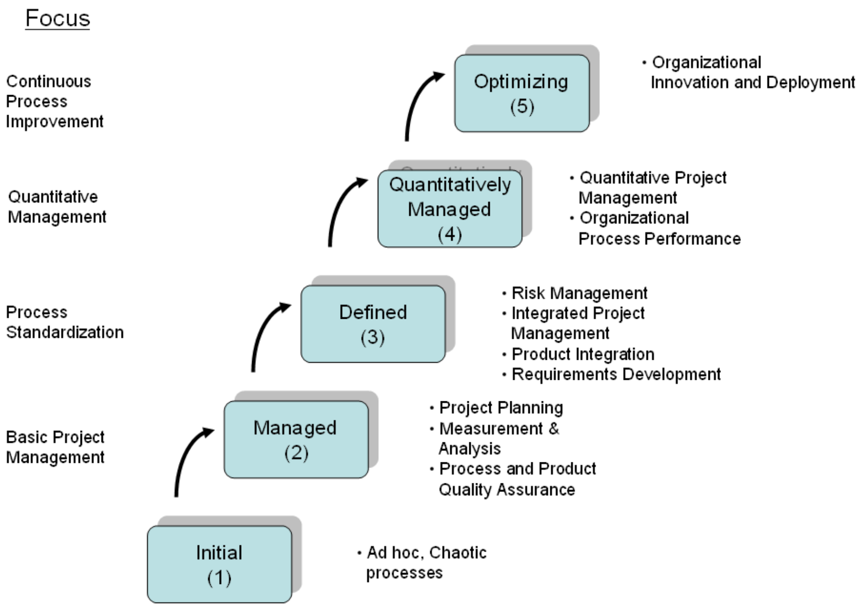

Focus
Continuous Process Improvement
Quantitative Management
Process Standardization
Basic Project Management
Optimizing (5)
Quantitatively Managed (4)
Defined (3)
Managed (2)
Initial (1)
• Organizational Innovation and Deployment
• Quantitative Project Management
• Organizational Process Performance
• Risk Management
• Integrated Project Management
• Product Integration
• Requirements Development
• Project Planning
• Measurement & Analysis
• Process and Product Quality Assurance
• Ad hoc, Chaotic processes

Figure 18: CMMI staged representation – maturity levels

Beside this two dispread models for software development process evaluation there exists some other approaches with the same goal, for example ISO/IEC 15504 (Software Process Improvement and Capability determination (SPICE)) [ISO/IEC98] [Drouin95] and BOOTSTRAP [KochKuvaja+94]. Since worldwide studies discovered that ISO 900x series and the CMMI are by far the most popular ones [KugRem95] [WangDorling+99], the other models are out if scope here.

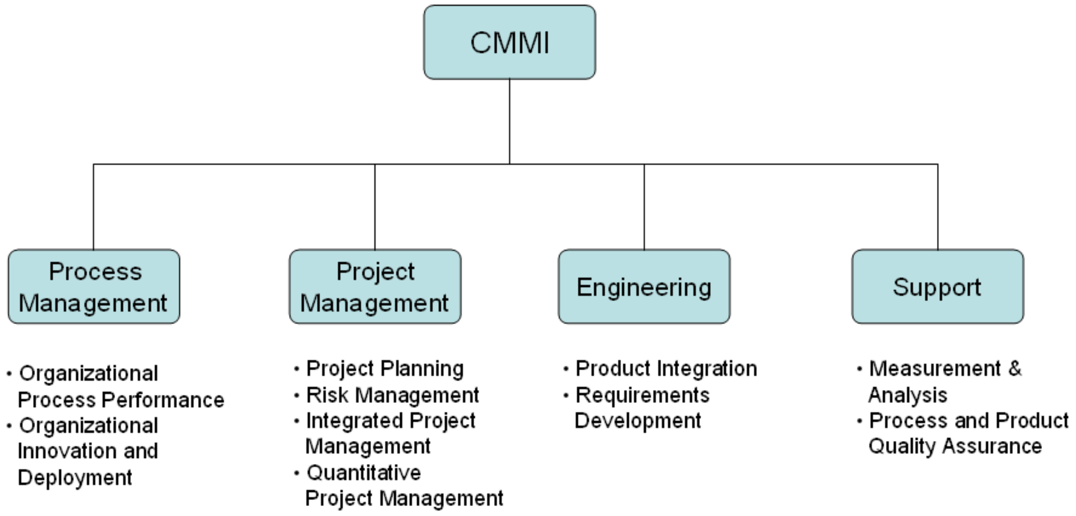

CMMI
Process Management
Project Management
Engineering
Support
• Organizational Process Performance
• Organizational Innovation and Deployment
• Project Planning
• Risk Management
• Integrated Project Management
• Quantitative Project Management
• Product Integration
• Requirements Development
• Measurement & Analysis
• Process and Product Quality Assurance

Figure 19: CMMI continuous representation – process areas

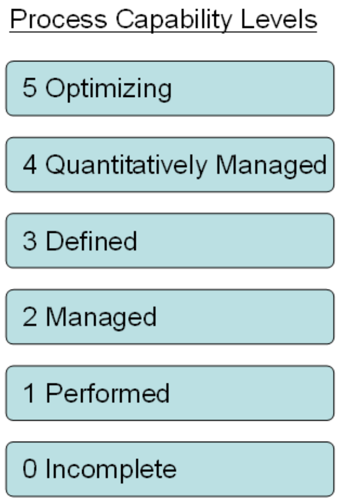


Figure 20: CMMI Continuous Representation – Process Capability Levels

The detailedness of the software process quality models and the definition of an optimum (e.g. CMMI Level 5) prescient these evaluation methods from software measures.

2.3.3. Measurement Process Methodologies

Integrated into a software development process the software measurement process aims to be a tool for the following goals [Ebert+05]:

- estimation of future product components, resources and process aspects
- analysis of artifacts, technologies and methods used in a development process
- structuring of the software process
- improvement of methodologies and techniques
- control the software development process and the resulting product quality

Consequently, different stakeholders or costumers of software measures can be identified as [Kueng00], [ListBruckner+05]:

- project management
- software engineers
- specialists (e.g. software quality management, process engineering, configuration management)
- functional (senior) management
- users/costumers

In order to aim the described goals for the named stakeholders different views on above described entities (product, resource, process) has to be distinguished.

In literature three major views for software measurement are defined. Strategic issues taken into account by functional (senior) management, tactical issues are addressed by project management, and technical issues are treated by specialists (e.g. software engineers or software quality managers) and costumers [Whitmire1997] [Ebert97].

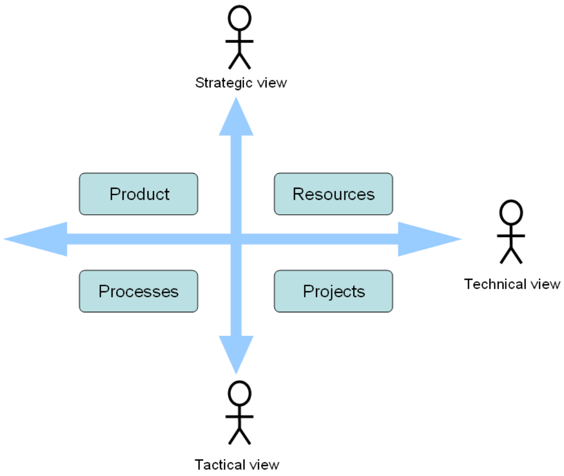


Figure 21: Views on software measurement

Strategic view

The strategic view is the most abstract one and addresses the feasibility of the whole business process in the long run. Therefore functional manager is interested in organizational goals which should be stated in measurable terms.

Usually the strategic view tracks trends of summarized statistics and determines if and how well business goals are being met. Since, not more than trend-setting data is used, raw or detailed software measures can be counterproductive.

Examples for such measures are:

- unit cost (labor hours/size)
- defect rates (delivered defects/size)
- cycle time (project days/size)

Tactical view

The tactical view is concerned about the performance of individual projects. Therefore project managers are responsible for established processes and they should use measurement data for building project planning and project controlling. In doing so measurement data is used to compare actual results to target results in which any variances should be investigated. Another use case for measurement values in the tactical view is the prediction of values by using measured ones (for example using project size to predict cost and schedule).

Examples for such measures are:

- Planning:
  - Resource (time and skill level) required for each task
  - Activities and processes required to complete tasks
  - Cost of resources required for the project
- Controlling
  - Actual effort-to-date by activity or product deliverable
  - Progress-to-date
  - Expected effort required to complete project
  - Target performance using target values of product measures
  - Defect data (defect rates, correction efficiency, cost of rework)

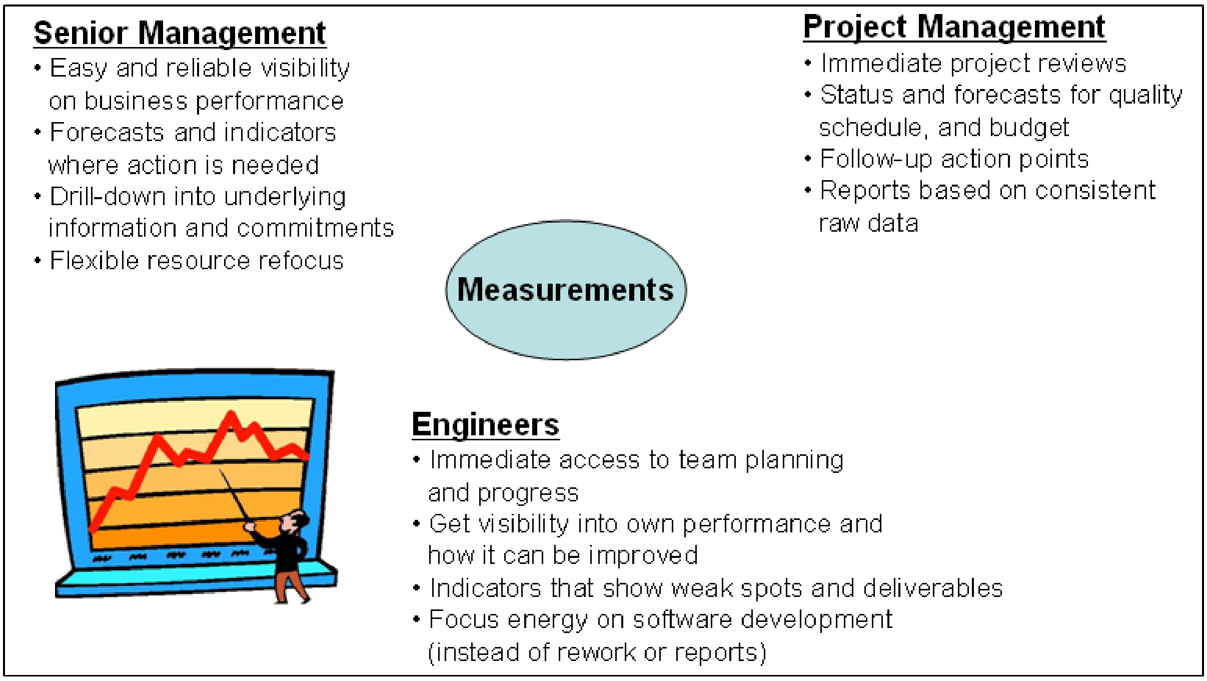
Technical view

All measures used in the strategic and technical views are built from fundamental technical measures. Physically, all previous described measurements are realized at the technical level. Furthermore the technical view deals with the detailed software measures needed for engineering purposes. They are focused upon a set of internal attributes of a single work product or process and the applied measures strongly depend on the used technology on the software development process.

Examples for such measures are:

- Size and complexity to choose between alternative implementations
- Coupling and cohesion to evaluate software design
- Defect rates and direct product measures to investigate relationships
- Productivity of own process to assess the effects if using new tools or methods

The different views and resulting differences in information needs for the roles have been summarized in the following figure [Ebert^{+}05].



Senior Management
- Easy and reliable visibility on business performance
- Forecasts and indicators where action is needed
- Drill-down into underlying information and commitments
- Flexible resource refocus

Project Management
- Immediate project reviews
- Status and forecasts for quality schedule, and budget
- Follow-up action points
- Reports based on consistent raw data

Measurements

Engineers
- Immediate access to team planning and progress
- Get visibility into own performance and how it can be improved
- Indicators that show weak spots and deliverables
- Focus energy on software development (instead of rework or reports)

Figure 22: Information need by stakeholder role

The different types of information needs and the manifold forms of technology and their application into a software development process facilitate a huge amount of different measures. In the course of time an uncounted number of measures have been published. A decade ago 1500 have been counted and a quantity of a few thousand has been estimated.

The increasing economic relevance of software measurement for organizations cannot be neglected. But issues like complexity and missing traceability of measurement processes constitute the need for direction and guidance in this regard.

To establish a well defined measurement process describing the path from the information need to the execution of measure has been targeted to making software measurement a success.

2.3.3.1. The Goal-Question-Metric Method

Researchers and practitioners have been trying over time to address this topic and to develop methodologies and procedures for a standardized measurement process. Early approaches propose a top-down architecture containing planning and execution of a set of software measures assigned to defined measurement goals.

One of the first models for a structured measurement process was published in the nineteen eighties by Basili et al. [BasiliWeiss84] [BasiliSelby^{+}86]. The *Goal/Question/Metric* (GQM) method assists one in detecting matching measurement approaches and measurement goals. (cf. [SolBer99])

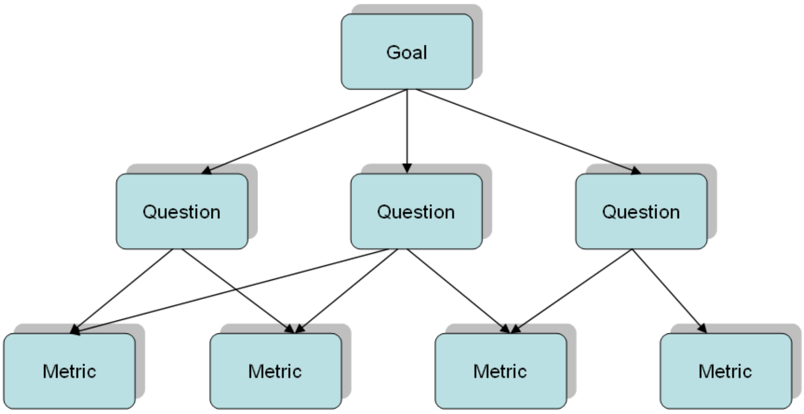


Figure 23: Goal-Question-Metric paradigm

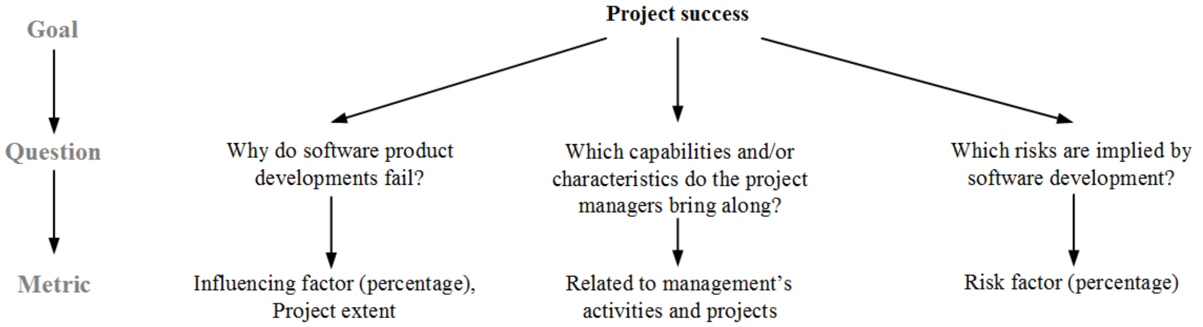

Goal
Question
Metric
Project success
Why do software product developments fail?
Which capabilities and/or characteristics do the project managers bring along?
Which risks are implied by software development?
Influencing factor (percentage), Project extent
Related to management's activities and projects
Risk factor (percentage)

Figure 24: An example of the appliance of the GQM method

GQM (as exemplified by [Dumke03]) initially requires the definition of distinct goals and in the following the compilation of suitable questions, which are thought to highlight the indicated issue from different sides. At the same time in the majority of cases hypotheses are nominated, that have to be approved or confuted in the course of the numerical evaluation. In case those hypotheses can be debilitated as wrong assumptions, they are to be classified among experts as myths. One of these wrong assumptions is for instance the one every programming expert being an excellent team leader, as well [Dumke03]. Ultimately, for the quantifiable answer to the questions only the required measures have to be determined.

In order to support the measurement beyond the identification of measures and to integrate the different sub processes the GQM approach was enhanced with a more holistic process model [GresseHoisl^{+}95].

This model separates the measurement process into four different phases (sub processes):

1. Planning
    a. establish GQM team
    b. create project plan

c. training of staff

2. Definition

   a. Definition of measurement goals

   b. Define Question and hypotheses

   c. Define metrics

   d. Review metrics on completeness and consistency

   e. Produce measurement plans

3. Data collection

   a. Create measurement database

   b. Define analysis methods

   c. Collect data

4. Interpretation

   a. Provide feedback

   b. Mapping of results to definition phase

   c. Control achievement of goals

   d. Report measurement results

The interrelations and connections between the particular phases are depicted in Figure 25.

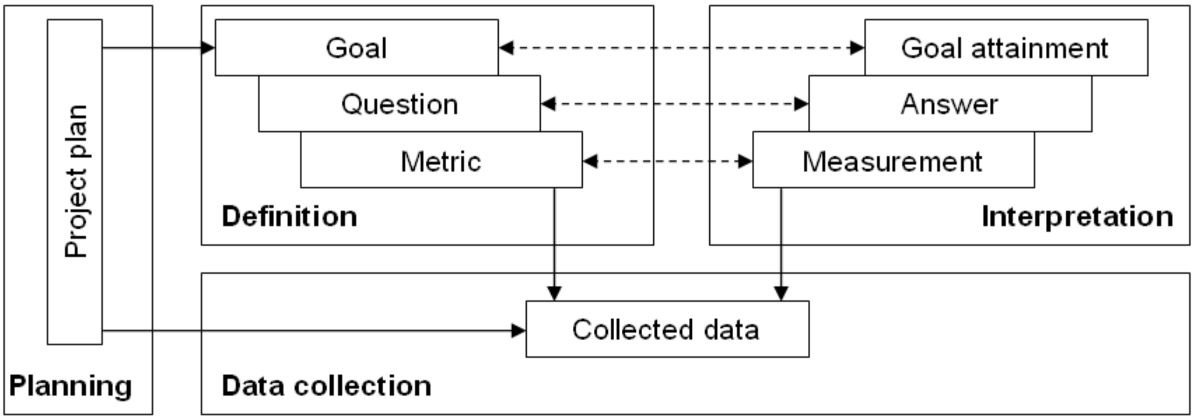


Figure 25: Goal-Question-Metric method

An often mentioned problem of goal oriented approaches is that practitioners feel not being involved in the goal definition and interpretation of measures which they have to implement.

But the straightforward and intuitive characteristic and a quite high propagation makes it a promising approach in the past.

A related approach can be found in the factor-criteria-metric model [Balzert08] where a quality goal is expressed by different factors. Afterward criteria's for the compliance of factors are defined and quantified by selected metrics.

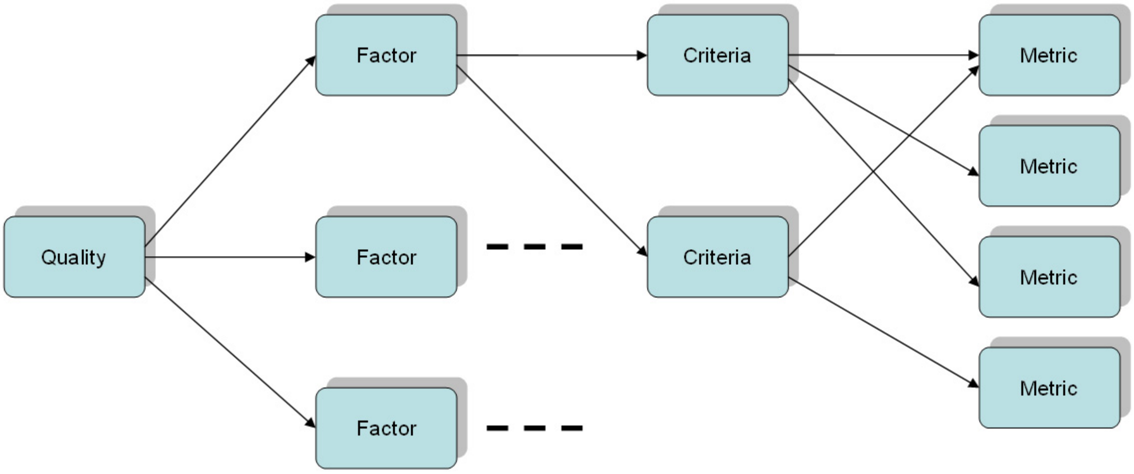


Figure 26: Factor-Criteria-Metric model

2.3.3.2. The E4 Software Measurement Process

But in the course of time process improvement gains more and more importance and the measurement process was surveyed in this regard, too.

In doing so the design of cyclic models has been identified as a promising approach for future measurement processes. Especially the idea of improvement objectives behind measured artifacts promotes the development of cyclic measurement processes.

One example for a cyclic measurement process is the E4 process model proposed by Ebert and Dumke containing four sub processes: "Establish", "Extract", "Evaluate", and "Execute".

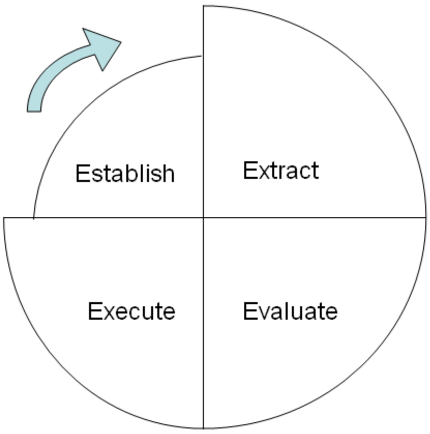


Figure 27: Continuous improvement circle [Ebert+05]

Trying to overcome described disadvantages of single ad-hoc measurement approaches two things can be identified as success factors for a successful measurement process [Ebert$^+$05]:

- integration of the measurement process into engineering and management processes
- fertilization of measurements by the corporate culture

Beside this integration perspective the realization of cyclic measurement approaches leads to a so called measurement lice-cycle. This aims to overcome single ad-hoc measurement activities.

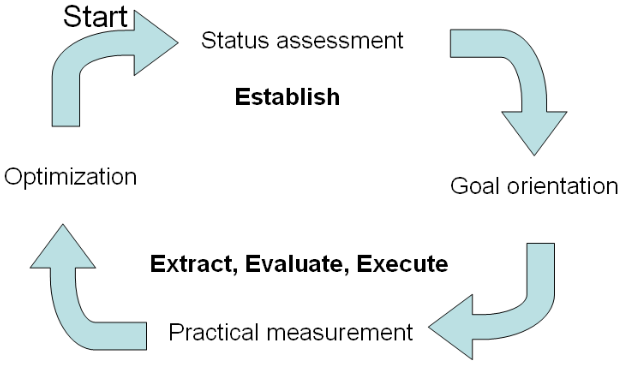


Figure 28: measurement life-cycle [Ebert$^+$05]

The measurement life-cycle contains four phases:

1. Status assessment: define improvement goals for enterprise and identify what matters.
2. Goal orientation: implementation of goal-oriented measurement.
3. Practical measurement: execution of software measurement.
4. Optimization: close the loop and provide feedback, decision for further measurement cycles.

2.3.3.3. The ISO/IEC15939 Software Measurement Process

Aiming the goals, the integration of measurement in project management and the realization of a cyclic measurement model, the ISO/IEC 15939 standard for software measurement has been created in 2002 [ISO/IEC02]. The standard contains four process categories: “Establish and sustain measurement commitment”, “Plan the measurement process”, “Perform the measurement process”, and “Evaluate measurement”. As a trigger for measurement activities it includes a process area “Technical and management processes”.

The activities are sequenced in an iterative cycle allowing for continuous feedback and improvement of the measurement process. Two activities are considered to be the Core

Measurement Process: Plan the Measurement Process, and Perform the Measurement Process. These activities mainly address the concerns of the measurement user. The other two activities provide a foundation for the Core Measurement Process, and provide feedback: Establish and Sustain Measurement Commitment and Evaluate Measurement. Measurements should be evaluated in terms of the added value they provide for the organization, and only deployed where the benefit can be identified. These later two activities address the concerns of the measurement process owner.

The layout of the processes and the included regularities are fully consistent with the CMMI process area "Measurement and Analysis".

The ISO/IEC 15939 Software Measurement Process helps to identify, define, select, apply, and improve software measurement in any software development project. This standard describes a compliant measurement process concerning its purposes and outcomes combined with appropriate activities and tasks.

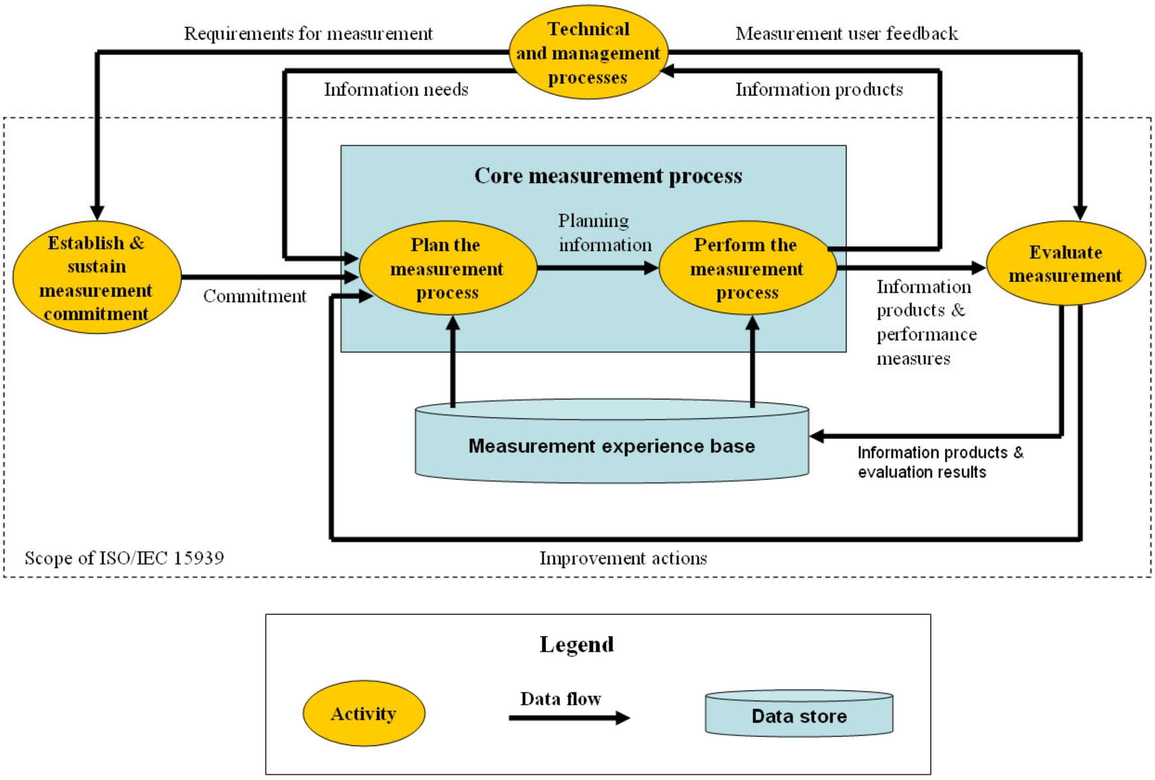

Requirements for measurement
Technical and management processes
Measurement user feedback
Information needs
Information products
Core measurement process
Planning information
Establish & sustain measurement commitment
Plan the measurement process
Perform the measurement process
Evaluate measurement
Commitment
Information products & performance measures
Measurement experience base
Information products & evaluation results
Scope of ISO/IEC 15939
Improvement actions
Legend
Activity
Data flow
Data store

Figure 29: The ISO/IEC 15939 software measurement process model [ISO/IEC02]

The basic tasks and artifacts/objects are:

establishingTasks = {acceptanceOfMeasurement, assignTheResources}

establishingArtefacts = {managementCommitment, measurementRequirements, planForMeasurementResources, descriptionOfTheOrganisationalUnit} *planningTasks* = *{organisationCharacterization, needsIdentification, measuresSelection, evaluationProcedures, evaluationCriteria, measurementResourcing, measurementSupporting}*

planningArtifacts = *{informationNeeds, descriptionOfTheOrganisationalUnit, candidateMeasures, selectedMeasures, measurementTasks, informationProducts, toolsDescriptions, trainingCourseMaterials}*

performingTasks = *{processIntegration, dataCollection, dataAnalysis, userCommunication}*

performingArtifacts = *{measuredArtifacts, collectedData, storedData, informationProducts, measurementExperienceBase}*

evaluationTasks = *{measurementEvaluation, measurementImprovement}*

evaluationArtifacts = *{evaluationCriteria, informationProducts, lessonLearned, measurementExperienceBase}*

explorationTasks = *{measurementReporting, measurementExploration, measurementConclusion}*

explorationArtifact = measurementExperienceBase

controllingTasks = {measurementFeedback, informationGeneration}

controllingArtifacts = {informationProducts, informationNeeds}

Mapping the formal description of the software measurement process to the ISO/IEC 15939 measurement standard one can assign the elements as shown in Figure 30.

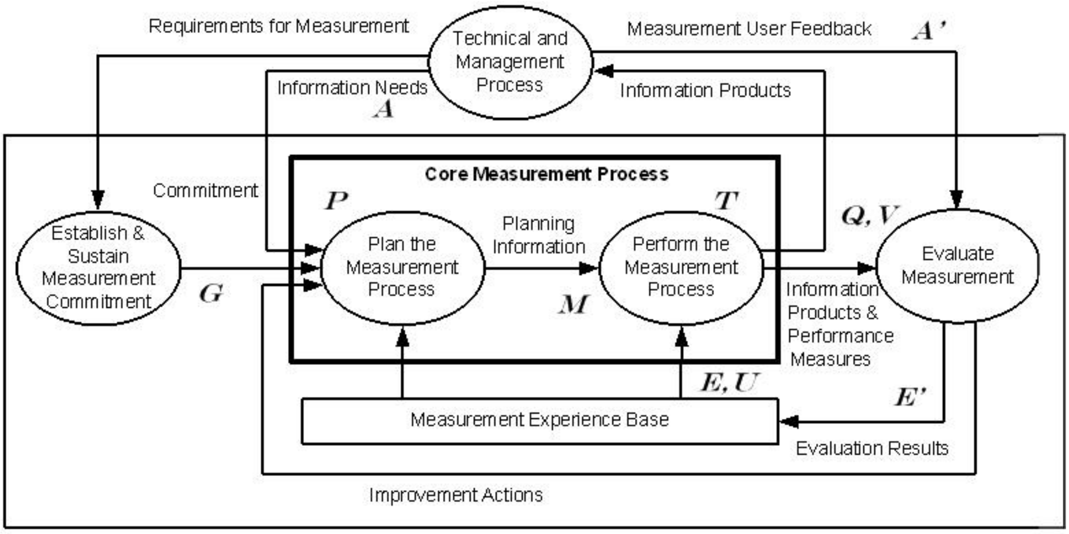

Requirements for Measurement
Technical and Management Process
Measurement User Feedback
A'
Information Needs
A
Information Products
Core Measurement Process
Commitment
P
T
Q, V
Establish & Sustain Measurement Commitment
G
Plan the Measurement Process
Planning Information
M
Perform the Measurement Process
Information Products & Performance Measures
Evaluate Measurement
E, U
E'
Measurement Experience Base
Evaluation Results
Improvement Actions

Figure 30: Assignment of measurement process elements to ISO/IEC 15939

As described regarding the GQM and E4 Model mapping information needs with measurable attributes is of high interest in measurement processes. In ISO/IEC 15939 the relationship is defined in the Measurement Information Model. The Measurement Information Model contains Interpretation, analysis Model, Measurement Function, and Measurement Method aiming to connect an information need with one or more measurable attributes. This connection is called a measurable concept.

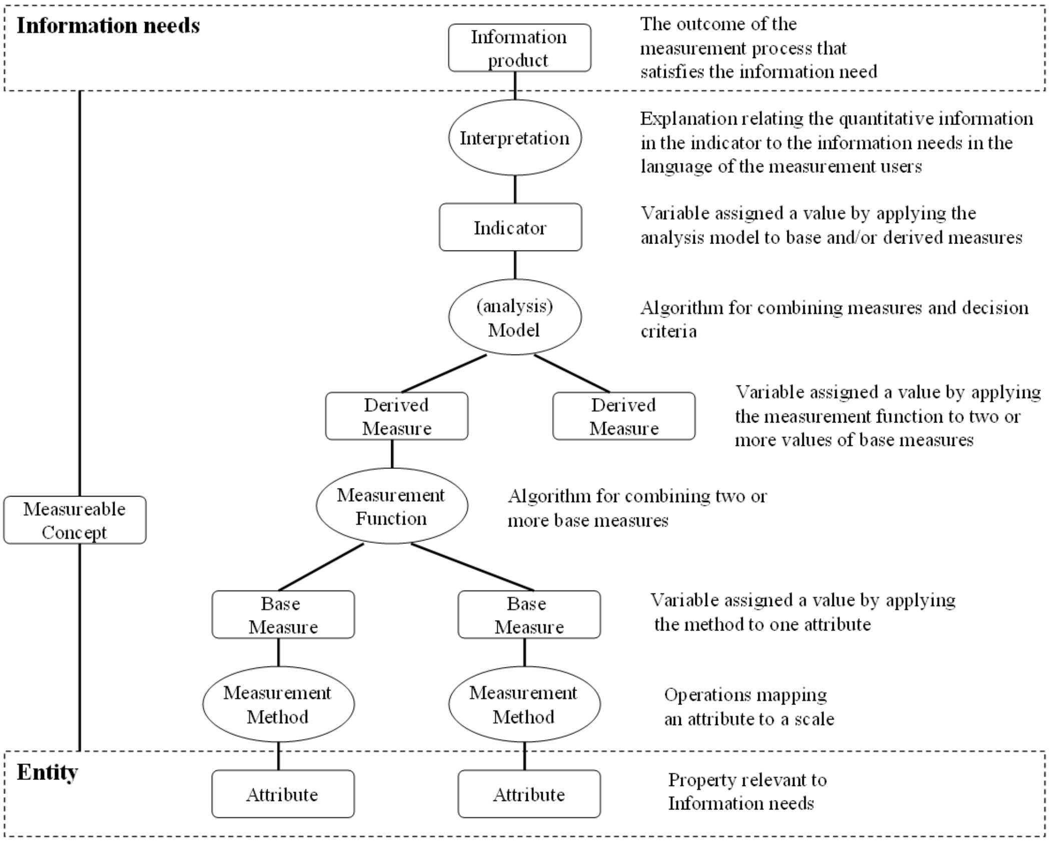

Information needs
Information product
The outcome of the measurement process that satisfies the information need
Interpretation
Explanation relating the quantitative information in the indicator to the information needs in the language of the measurement users
Indicator
Variable assigned a value by applying the analysis model to base and/or derived measures
(analysis) Model
Algorithm for combining measures and decision criteria
Derived Measure
Derived Measure
Variable assigned a value by applying the measurement function to two or more values of base measures
Measureable Concept
Measurement Function
Algorithm for combining two or more base measures
Base Measure
Base Measure
Variable assigned a value by applying the method to one attribute
Measurement Method
Measurement Method
Operations mapping an attribute to a scale
Entity
Attribute
Attribute
Property relevant to Information needs

Figure 31: Key relationships in the measurement information model

2.3.4. Software Measurement Programs

Mapping the theoretical foundations and concepts of the software measurement process to the practical application in software engineering industry has been aimed by using different approaches which can be recognized by the number of references reviewed.

Among others the terms “metrics program” or “measurement framework” are used to describe the application or implementation of software measurement processes in software engineering processes. [Kitchenham96] [PerkinsPeterson$^+$03] [Goodman04], [Mendonça97].

Out of this meaning and the definitions in Abran et al. [AbrLaf$^+$99], and Berry and Jeffery [BerryJeffery00], software measurement programs can be defined as follows:

“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."

A major driving force for establishing and sustaining a software measurement program are often legal terms or other contracts which dictate a specific process quality level and thereby the use and application of software measurement. Apart from that companies even, if they are convinced of the usefulness of a software measurement program, have to do a cost-benefit analysis to know whether or not the investment of personal and financial resources is worth. Experience reports regarding the cost of measurement programs points out that the overall cost for establishing such program should not exceed 10% of the overall project costs [GrableJernigan$^+$99]. The six major outcomes of successful measurement programs are [Solingen04]:

- Increase productivity
- Fewer defects
- Reduced development time
- Increase reusability
- Future expertise of personal resources
- Increase user satisfaction

The quantification of costs and benefits is not an easy task because some of the benefits are difficult to express in numbers and not implementing a measurement program can end up in failed projects and thereby risk the entire business.

To overcome this situation recent approaches try to measure these benefits and calculate the return on investment (ROI) [EweWag05] [Rico04] of software measurement programs and presents an average ROI of 8 [Solingen04]. The return on investment is thereby calculated by dividing a financial representation n of the resulting benefits by a financial representation of the costs.

Table 7: Return on Investment of software measurement programs

Context	**Return on Investment**
General Dynamic	2.2
US Navy	4.1
Hughes Aircraft	5
IBM Global Services India	5.5
Motorola	6.77

Philips	7.5
Raytheon	7.72
Boeing	7.75
Hewlett-Packard	10.4
Northrop Grumman	12.5
Odgen ALC	19
Average	**8.04**

Of course, this numbers should be used cautiously because usually only success stories are published. Taking into account other reports about tackled software measurement programs the actual situation is quite different:

According to Kaner and Bond [KanerBond04] just a few companies establishes a program and even less can finally succeed. Unfortunately those who come to pain are often kept secret. [HallFenton97] Already from 1988 Howard Rubin [Rubin87] [Rubin90] produced statistics about measurement program successes or failures. Also in the eighties, an industry survey [Hetzel90] has revealed that less than 10% of respondents had a positive image of measuring software because of bad experiences. Since then, these statistics show a permanent disability rate of 78% or more within two years of the program proper functioning elapsed. Another corroborating Desharnais study [Desharnais94] among the twenty Canadian companies have analyzed successes and failures in the implementation of SMP and exhibitions same failure rate 60%. This results could be clarified by a study from Dekkers [DekkersMcQuaid02] subsumed the investigations in this regard and comes up with a failure rate of almost 80% in the first two years after establishment on a population of up to 640 measurement programs in the industry. This study was reconsidered in 2004 and the quintessence as well as the statement about the success rate has been confirmed [McQuaidDekkers04].

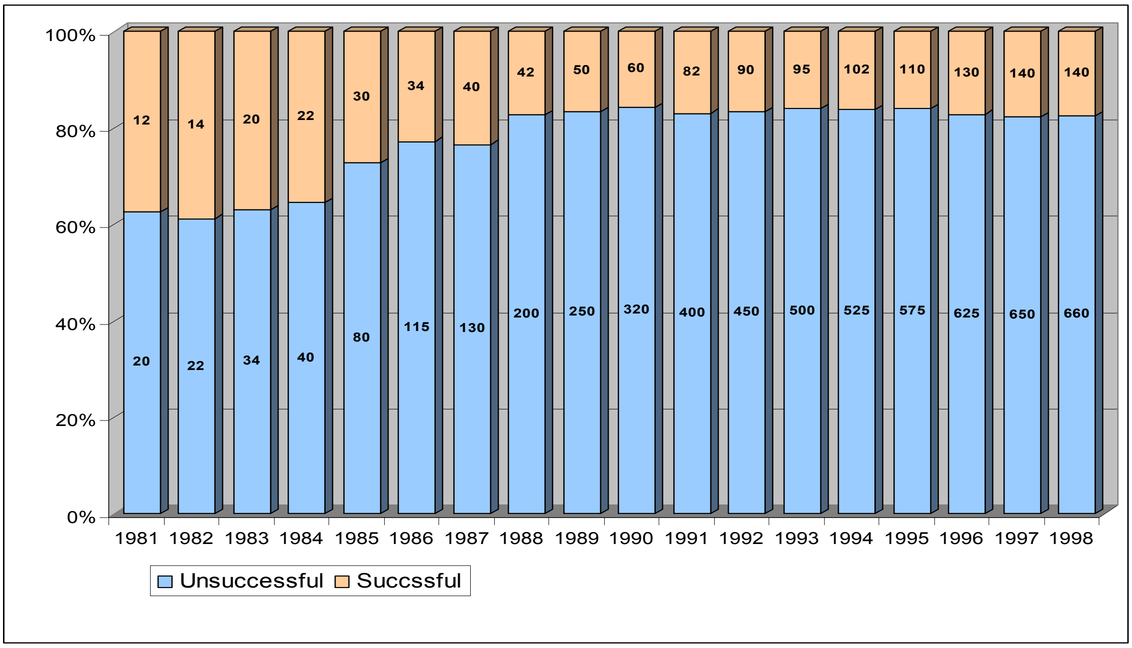


Figure 32: Results of established corporate measurement programs

Especially the fact that over time the success rate declines draws a worrying picture and the reasons as well as possible solutions should be investigated.

A very detailed analysis of pitfalls and best practices for implementation and sustainment of software measurement programs can be found in [Braungarten07].

Beside reasons related to software measurement in general:

- misunderstanding of measurement theory [DekkersMcQuaid02]
- lack of senior management's commitment [Dennis02]
- lack of ongoing education of measurement theory among staff [KanerBond04]

Some pitfalls are related to the used measurement tools:

- nonexistence or expensiveness of automated tool support [BakerHantos03]
- fear of staff against effort for tool usage and data analysis [Minkiewicz00] [deSilveira2002]
- unclear cost-benefit ratio of used tools [EmamBriand97][Reifer02]
- inflexible usage of existing measurement tools [Johnson01]

In the course of time experiences with software measurement programs lead to distinct best practices regarding measurement context, measurement process, measurement product, and measurement input. Important best practices are related to the measurement tools:

- easy to use measurement tools for automatic data collection and analysis [UmaEmu05] [Jones03] [Holmes02]

- real-time data collection and securing of data integrity and consistency [EbertDumke07][Russac02]
- storage of software measurement data into repositories or comprehensive databases [DekkersMcQuaid02] [Harrison04]
- general access to software measurement data for the collectors or stake holders [Pfleeger93]
- realistic assessment of cost-benefit ratios, ROI, or implementation costs [DawsonNolan03] [Kitchenham96]
- enable incremental implementation of software measurement programs (starting with a small set if high-priority measures) [EbertDumke07][Goodman04] [IveMat00]
- enable monitoring of trend analysis over time [Wiegers97]
- provide quick feedback on measurement results [Wiegers99][DekkersMcQuaid2005] [GrableJernigan+99]

Since it has been identified as a key factor for the success of a software measurement program the maturity of measurement technology is addressed in some of the above mentioned reasons [Kunz+06e]. To analyze the importance of measurement technology and to draw connection between different levels of measurement process maturity and measurement technology maturity, [Daskalantonakis94] proposed a maturity model for software measurement maturity assessment. Based on similar assumptions in other maturity model like CMMI (see chapter 2.3.2.2. CMMI Framework for Process Integration and Product Improvement) they propose six maturity levels. Later adoptions [BP95], [BP96a] and analysis [Braungarten07] of the maturity model ended up in the maturity grid presented in following table.

Table 8: Maturity grid for software measurement maturity assessment [Braungarten07]

Topic	Level 1 (initial)	Level 2 (repeatable)	Level 3 (defined)	Level 4 (managed)	Level 6 (optimized)
1. Formalization of the development process	- Process unpredictable - Project depends on experienced professionals - no/poor process focus	- Repeat mastered tasks - Process depends on experienced professionals	- Process characterized and understood	- Process measured and controlled	- Process optimized - Focus on process improvement
2. Formalization of the measurement process	- Little or no formalization	- Formal procedures established	- Standards applied	- improvement procedures established - internal standards applied	- Organization has been learned and improved

3. Scope of measurement	- Done occasionally on projects with experienced people or not at all	- Done on projects with experienced people - Project estimation techniques exists - Project focus	- Data collection and recording - Existence of specific automated tools - Product focus	- Metric packages applied - Existence of integrated automated tools - Process focus	- Organization 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 all projects	- Process database - corporate database containing process information	- Knowledge base - Improvement and learning database
5. Measurement evaluation	- Little or no measurement conducted	- Project measurement and management	- Product measurement and management	Process measurement and management	- Continuous measurement feedback and improvement
6. Measurement support for management control	-Management not supported by measurement	- Little support by measurement - Basic control	- Product measurement and control	- Management process measured and controlled	- Can define technology values and needs - can check achievements
7. Project improvement	- No statistical process control - No senior management involvement	- Disciplined project and configuration management - Risk management - Subcontractors evaluated	- Dedicated process resources - Process data not retained nor analyzed - 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 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 determine problem causes	- Can analyze and prevent problem causes
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 - Staff training established	- 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 optimization - Can analyze and prevent problem causes

Summary: Software Measurement Systems and Processes

The classification of the measurement process *MP* itself is based on the measured artifact. The measurement of aspects (aspects product or processes or resources) leads to the *aspect-oriented measurement*. The measurement of all aspects of a product or all aspects of the process or all aspects of the resources would be called as *capability-oriented measurement*. If we involve all software artifacts (product and process and resources) we will call this as a *whole measurement*. These characteristics build the "approach" attribute of measurement process.

The inclusion of measurement activities (e.g. measurement and analysis, effort estimation) itself confirm the exceptional position and constitute that software measurement becomes part of a software engineering process model.
In this way one can identify a software measurement process which can potentially use every above mentioned measure to satisfy an information need for the management of a software engineering process.

Otherwise, the "solution" characteristic of the measurement process can be explained depending on their kind of performing such as *in-house* or *outsourced* or based on methodology of *global production*.

Unfortunately, there is no general international consensus about a defined set of measurement activities for the development of software. Because of that the implementation of above described theoretical foundations, concepts, processes, and standards as a Corporate Measurement Program seems to be a promising approach and increasingly important. Though, it does require taking into account manifold aspects and can be considered as non-trivial. [FentonPfleeger97], [Goodmann93]

Based on the identified best practices for SMP implementation two areas for the support and assistance for successful SMP can be identified: infrastructures for measurement consisting out of different tools and the automation of software measurement (sub) processes.

For the identification of a better solution regarding both aspects the current situation in the tool area should be considered taking into account both goals: acceptance of the stakeholders and automated procedures.

2.4. Software Measurement Paradigms

As outlined in the previous sections of this chapter the application of software measurement can be done using different ways and techniques of quantifying software entities. To characterize the existing ones and to draw the connections this chapter is intended to provide a detailed consideration of measurement paradigms.

2.4.1. Basics of Scalability

In this section a first graduation of the software measurement characteristics is introduced. The idea of classification of measurement aspects and processes is not new. Examples are

1. Zelkowitz defines a ranking of validation of research papers as a 14-scale taxonomy in decreasing manner as: *project monitoring, case study, field study, literature search, legacy data, lessons learned, static analysis, replicated experiment, synthetic, dynamic analysis, simulation, theoretical, assertion, no experimentation* [Zelkowitz07].

2. A consideration of the experiment levels by Kitchenham leads to (also decreasing): *industrial case studies, quasi experiment*, and *formal experiment* [Kitchenham07].

3. Sneed identifies a ranking of (function point based) productivity related to the kinds of developed systems as (decreasing): *industry, trading, governance, assurance* and *banking* [Sneed05].

For the considerations, experiences and some of the results from industrial projects at Alcatel, Siemens, Bosch, and German Telecom ([BraKun+05], [Dumke07], [EbertDumke07], [Richter05], [SchmKunz+07]) is used in order to achieve a *holistic approach*. The different aspects of the measurement process component are defined as a first assumption in an *ordinal manner/scale* (considering also [BourqueOligny+07], [Braungarten07], [FarBra+05], [LairdBrennan06], [Pandian04], [SchmKunz+07], and [Sneed05]). First ordinal classifications of the measurement process components in an increasing manner are the following

G: *intention* ∈ {*understanding, evaluation, improving, managing*}

viewpoint ∈ {*internal_goals, external_goals, goals_in_use*}

A: *domain* ∈ {(*product_aspects* ∨ *process_aspects* ∨ *resources_aspects*),

(*product* ∨ *process* ∨ *resources*), (*product* ∧ *process* ∧ *resources*)}

origin ∈ { *other_pendant, pendant_in_same_domain, original* }

M: *method* ∈ {*experiment/case study, assessment, improvement, controlling*}

sort ∈ {*analogical_conclusion, estimation, simulation, measurement*}

T: *level* ∈ {*manual, semi-automatic, automatic*}

support ∈{*one_measurement_phase,some_measurement_phases, whole_measurement*}

P: *kind* ∈ {*manager, researcher, practitioner*}

area ∈ {*measurement_expert_staff, measurement_application_staff*}

Q: *value* ∈ {*identifier/nomination, ordinal_scale*} *structure* $\in$ {*single_value*, (*normalization* $\vee$ *transformation*), *aggregation*}

E: *form* $\in$ {(*intuition* $\vee$ *law* $\vee$ *trend* $\vee$ *principle*), *analogy*, (*criteria* $\vee$ *rules_of_thumb*), (*axiom* $\vee$ *lemma* $\vee$ *formula*)}

contents $\in$ {(*limits* $\vee$ *threshold*), (*gradient* $\vee$ *calculus*), *proof*}

V: *measure* $\in$ {*interval_scale*, *ratio_scale*}

aggregation $\in$ {*values*, (*data_basis* $\vee$ *repository*), (*dashboard* $\vee$ *cockpit*)}

U: *type* $\in$ { *sociological_unit*, *economical_unit*, *physical_unit*, *hardware_unit*, *software_unit*}

standard $\in$ {*non_standard*, *quasi_standard*, *standardized*}

E': *form:* see above

extension $\in$ {*correction*, (*refinement* $\vee$ *approximation* $\vee$ *adaptation*), *extension*}

A': *domain:* see above

changing $\in$ { *understood*, *improved*, *managed*, *controlled* }

Including the different levels of performing the measurement in the IT area leads to the following classification

MP: *approach* $\in$ {*aspect-oriented_measurement*, *capability-oriented_measurement*, *whole_measurement*}

solution $\in$ { *outsourced*, *global_production*, *inhouse*}

Note that the exponents address the main characteristics and the indexes show the sub characteristics. This assumption explains some first relationships.

2.4.2. Main Characteristics Preferences of Measurement Process Components

In the following some examples of this kind of measurement aspects scaling are presented. Related to the measurement artifacts the following elements are established (note that the sign "$\precsim$" characterizes the so-called ***evidence level*** (see [Kitchenham07]):

$$A_{origin}^{aspects} \precsim A_{origin}^{product \vee process \vee resources} \precsim A_{origin}^{product \wedge process \wedge resources} .$$

Considering the measurement and including the application leads to

$$M_{sort}^{case\_study} \precsim M_{sort}^{assessment} \precsim M_{sort}^{improvement} \precsim M_{sort}^{controlling} .$$

Addressing the tool aspects gives

$$T^{\overset{manual}{support}} \precsim T^{\overset{semi\_automatic}{support}} \precsim T^{\overset{automatic}{support}} .$$

Achieving the personnel background it is obtained:

$$P^{\overset{manager}{area}} \precsim P^{\overset{researcher}{area}} \precsim P^{\overset{practitioner}{area}} .$$

And finally addressing the used experiences leads to

$$E^{\overset{principle}{contents}} \precsim E^{\overset{analogy}{contents}} \precsim E^{\overset{rules\_of\_t\ humb}{contents}} \precsim E^{\overset{formula}{contents}} .$$

2.4.3. Sub Characteristics Preferences of Measurement Process Components

Considering the sub characteristics chosen relationships are given:

$$A^{\overset{domain}{other\_pendant}} \precsim A^{\overset{domain}{domain\_pendant}} \precsim A^{\overset{domain}{original}} .$$

Describing the measurement and application aspects obtains

$$M^{\overset{method}{analogical\_conclusion}} \precsim M^{\overset{method}{estimation}} \precsim M^{\overset{method}{simulation}} \precsim M^{\overset{method}{measurement}} .$$

Relating the tool aspects leads to

$$T^{\overset{level}{one\_meas.\_phase}} \precsim T^{\overset{level}{some\_meas.\_phases}} \precsim T^{\overset{level}{whole\_measurement}} .$$

Achieving the personnel background as

$$P^{\overset{kind}{measurement\_expert}} \precsim P^{\overset{kind}{applicatio\ n\_staff}} .$$

Furthermore, considering the experiences it is defined:

$$E^{\overset{form}{threshold}} \precsim E^{\overset{form}{gradient}} \precsim E^{\overset{form}{proof}} .$$

2.4.4. Combined Characteristics Preferences of Measurement Process Components

Finally, using both kinds of characteristics leads to the following example relationships.

$$A^{\overset{aspects}{domain\_pen\ dant}} \precsim A^{\overset{aspects}{original}} \precsim A^{\overset{product \vee process \vee resources}{other\_pend\ ant}}$$

$$\precsim A_{\mathit{domain\_pendant}}^{\mathit{product} \vee \mathit{process} \vee \mathit{resources}} \precsim A_{\mathit{other\_pendant}}^{\mathit{product} \wedge \mathit{process} \wedge \mathit{resources}} .$$

or

$$\precsim M_{\mathit{analogical\_conclusion}}^{\mathit{case\_study}} \precsim M_{\mathit{estimation}}^{\mathit{improvement}} \precsim M_{\mathit{estimation}}^{\mathit{controlling}} \precsim M_{\mathit{simulation}}^{\mathit{experiment}}$$

$$\precsim M_{\mathit{simulation}}^{\mathit{assessment}} \precsim M_{\mathit{measurement}}^{\mathit{case\_study}} \precsim M_{\mathit{measurement}}^{\mathit{assessment}} \precsim M_{\mathit{measurement}}^{\mathit{controlling}} .$$

and

$$E_{\mathit{threshold}}^{\mathit{law}} \precsim E_{\mathit{calculus}}^{\mathit{law}} \precsim E_{\mathit{proof}}^{\mathit{law}} \precsim E_{\mathit{limits}}^{\mathit{analogy}} \precsim E_{\mathit{gradient}}^{\mathit{analogy}} \precsim E_{\mathit{proof}}^{\mathit{analogy}}$$

$$\precsim E_{\mathit{threshold}}^{\mathit{criteria}} \precsim E_{\mathit{calculus}}^{\mathit{rules\_of\_thumb}} \precsim E_{\mathit{threshold}}^{\mathit{axiom}} \precsim E_{\mathit{gradient}}^{\mathit{lemma}} \precsim E_{\mathit{proof}}^{\mathit{formula}} .$$

2.4.5. Simplified Examples of Measurement Process Description

At first the formal descriptions is used in order to describe some typical software measurement situations and implementations. Therefore some different levels of ***measurement evidence*** such as

- Using only the next lower levels of previous paradigm measurement experiences leads to the ***measurement approximation***
- Using one or more of the second and/or third lower substitution levels can be considered as ***measurement qualification***
- Using only the lowest level of previous paradigm measurement experiences leads to the ***measurement initialization***

are established.

In the following some examples using the scaled measurement process description are described. Usually, in the software development and application one can describe some of the following tasks and activities based on our formal background [DumBra+07].

First general metrics application:

A first example shows a simple application of metrics based on a basic measurement process.

$$MP_{\mathit{inhouse}}^{\mathit{aspect\_oriented}} :$$

$$(G_{\mathit{internal\_goals}}^{\mathit{evaluation}} \times A_{\mathit{original}}^{\mathit{product\_aspects}} \times M_{\mathit{measurement}}^{\mathit{experiment}})_{T_{\mathit{some\_meas.\_phases}}^{\mathit{semi\_automatic}},\, P_{\mathit{measurement\_expert}}^{\mathit{practitioner}}}$$

$$\rightarrow (Q^{\text{ordinal\_scale}}_{\text{normalization}} \times E^{\text{formulas}}_{\text{threshold}})$$

Product quality assurance:

Then next example describes a more practical situation considering the (full) product measurement in an IT area.

$$MP^{\text{capability}-\text{oriented}}_{\text{inhouse}} \quad :$$

$$(G^{\text{managing}}_{\text{external\_g oals}} \times A^{\text{product}}_{\text{original}} \times M^{\text{assessment}}_{\text{measuremen t}})_{T^{\text{semi\_autom atic}}_{\text{measuremen t\_phases}}, P^{\text{practitioner}}_{\text{measurement\_expert}}}$$

$$\rightarrow V^{\text{ratio\_scale}}_{\text{cockpit}} \times U^{\text{software}}_{\text{standardized}}$$

Process improvement:

This example characterizes some of the process improvements using process improvement standards.

$$MP^{\text{capability}-\text{oriented}}_{\text{inhouse}} \quad :$$

$$(G^{\text{improving}}_{\text{goals\_in\_u se}} \times A^{\text{process}}_{\text{original}} \times M^{\text{improvement}}_{\text{measurement}})_{T^{\text{semi\_automatic}}_{\text{measurement\_phases}}, P^{\text{practitioner}}_{\text{measuremen t\_appl.\_staff}}}$$

$$\rightarrow (Q^{\text{ordinal\_sc ale}}_{\text{aggregatio n}} \times E^{\text{criteria}}_{\text{threshold}})_{T^{\text{semi\_automatic}}_{\text{measurement\_phases}}, P^{\text{practitioner}}_{\text{measurement\_expert}}}$$

$$\rightarrow E^{\text{criteria}}_{\text{approximat ion}} \times A^{\text{process}}_{\text{improved}}$$

Project controlling:

Another example of process measurement and evaluation is given in the following.

$$MP^{\text{capability }-\text{oriented}}_{\text{global\_pro duction}} \quad :$$

$$(G^{\text{managing}}_{\text{external\_g oals}} \times A^{\text{process}}_{\text{original}} \times M^{\text{controlling}}_{\text{measurement}})_{T^{\text{automatic}}_{\text{whole\_measurement}}, P^{\text{practictio ner}}_{\text{meas.\_appl .\_staff}}}$$

$$\rightarrow (V^{\text{ratio\_scal e}}_{\text{cockpit}} \times U^{\text{software\_unit}}_{\text{standardized}})_{T^{\text{automatic}}_{\text{whole\_meas urement}}, P^{\text{practictioners}}_{\text{meas.\_appl\_staff}}}$$

$$\rightarrow E^{\text{criteria}}_{\text{adaptation}} \times A^{\text{process}}_{\text{controlled}}$$

Resources adaptation:

The last example is addressed to the resource measurement as an improvement of the IT infrastructure.

$$MP^{capability-oriented}_{outsourced} :$$

$$(G^{improving}_{goals\_in\_u\ se} \times A^{resources}_{pendant} \times M^{improvemen\ t}_{measuremen\ t})_{T^{semi\_automatic}_{measurement\_phases},P^{practition\ er}_{measuremen\ t\_expert}}$$

$$\rightarrow (Q^{identifica\ tion}_{sinle\_valu\ e} \times E^{intuition}_{threshold})_{T^{semi\_automatic}_{measurement\_phases},P^{practition\ er}_{measuremen\ t\_expert}}$$

$$\rightarrow E^{analogies}_{adaptation} \times A^{resources}_{improved}$$

These examples demonstrate some of the possible constellations of measurement processes. One example involves an aspect-oriented approach and the other ones are capability-oriented. In order to perform a general comparison and classification one must consider all the *MP* characteristics (at first the *G* level then the *A* level etc.). That leads to

$$MP^{traditional}_{first\_metrics\_appl.} \precsim MP^{traditional}_{process\_improvement} \precsim MP^{traditional}_{resource\_adaptation} \precsim$$

$$MP^{traditional}_{product\_quality\_assurance.} \precsim MP^{traditional}_{project\_controlling}$$

This is only one of the results. On the other hand one can identify the point of view in order to achieve any improvement in the measurement process level.

2.4.6. Measurement Process Improvements

In the sections above an ordinal scaled multi-dimensional “space” of software measurement aspects have been characterized that consists of the ***lowest measurement level*** as

$$MP^{aspect\_ori\ ented}_{outsourced} :$$

$$(G^{understanding}_{internal\_goals} \times A^{product}_{pendant\_in\_same\_domain}$$

$$\times M^{experiment}_{analogical\_conclusion})_{T^{manual}_{one\_meas.\_\ phases},P^{manager}_{meas.\_expert\_staff}}$$

$$\rightarrow (Q^{identification}_{sinle\_value} \times E^{intuition}_{threshold})$$

some ***immediate levels*** or measurement situations such as

$$MP^{\text{aspect\_oriented}}_{\text{inhouse}}:$$

$$(G^{\text{evaluation}}_{\text{external\_goals}} \times A^{\text{product\_aspects}}_{\text{original}} \times M^{\text{assessment}}_{\text{estimation}})_{T^{\text{semi\_automatic}}_{\text{some\_meas.\_phases}}, P^{\text{researcher}}_{\text{meas.\_appl.\_staff}}}$$

$$\left\{ \begin{array}{c} (Q^{\text{nomination}}_{\text{normalization}} \times E^{\text{analogy}}_{\text{calculus}}) \\ \\ V^{\text{interval\_scale}}_{\text{data\_basis}} \times U^{\text{hardware\_unit}}_{\text{quasi\_standard}} \end{array} \right\}$$

(can be improved by “*aspect-oriented*” → “*capability-oriented*”, “*evaluation*” → “*improving*”, “*external_goals*” → “*goals_in_use*”, “*product_aspect*” → “*product*” etc.)

and the ***highest software measurement level***

$$MP^{\text{whole\_measurement}}_{\text{inhouse}}:$$

$$(G^{\text{managing}}_{\text{goals\_in\_use}} \times A^{\text{product} \wedge \text{process} \wedge \text{resources}}_{\text{original}} \times M^{\text{controlling}}_{\text{measurement}})_{T^{\text{automatic}}_{\text{whole\_measurement}}, P^{\text{practitioner}}_{\text{meas.\_appl.\_staff}}}$$

$$\rightarrow (V^{\text{ratio\_scale}}_{\text{cockpit}} \times U^{\text{software\_unit}}_{\text{standardized}})_{T^{\text{automatic}}_{\text{whole\_measurement}}, P^{\text{practitioner}}_{\text{meas.\_appl.\_staff}}}$$

$$\rightarrow E^{\text{formulas}}_{\text{extension}} \times A^{\text{poduct} \wedge \text{process} \wedge \text{resources}}_{\text{controlled}}$$

Summary: Measurement Paradigms and Improvements

Finally, the following ***graduation of measurement improvements*** is presented as a first kind of improvement classification:

- ***Weak measurement improvement***: This kind of improvement consists of an improvement of a *measurement sub characteristic* to the next level (as one step).
- ***Moderate measurement improvement***: The improvement of the measurement process based on *more than one step of a/some sub characteristic(s)* building this kind of measurement process improvement.
- ***Essential measurement improvement***: This kind of improvement consists of an improvement of a *measurement main characteristic* to the next level (as one step).
- ***Remarkable measurement improvement:*** The improvement of the measurement process based on *more than one step of a/some main characteristic(s)* building this kind of measurement process improvement.

Therefore, based on the formal described measurement process methods of measurement improvement are identified easily.

3. Performing the Software Measurement Process

"The efficiency of the software measurement process depends on the level of automation by tools."*[Ebert+05]*

Tool support is an important factor in every area of software engineering and ensures activities starting with the requirement definition at the beginning until software test and maintenance at the end of the software development life-cycle.

Therefore the purpose is to determine the performance or quality of the software product or the software development process by producing and collecting measurement data by so-called *measurement tools* [Dumke05]. The need for a measurement tool application and thereby the support of measurement activities strongly enhances the acceptance of software measurement methods. It is a well known and commonly accepted coherency that one can't expect that a measurement method will be really accepted by users if there is no adequate tool support [BunDek08], [Lother2007], [Dumke96], [KnöllBusse91] [Kunz+08e].

For the different intentions the notion of computer-assisted software measurement and evaluation (CAME) can be used to identify the different kinds of measurement tools [DumGri96]. In this regard the usage of a measurement tool is based on define measurement frameworks [Dumke96], [OmanPfleeger96], [Zuse98] and they can be classified according to the degree if integration on software development environments for instance as integrated forms, external coupling forms or stand-alone, monolithic measurement tools [EbertDumke07].

Taking into account the different software measurement phases the scope of measurement tools are identified according to Figure 33. The dotted lines represent typical stage coverage by single tools (adapted from [Dumke05]).

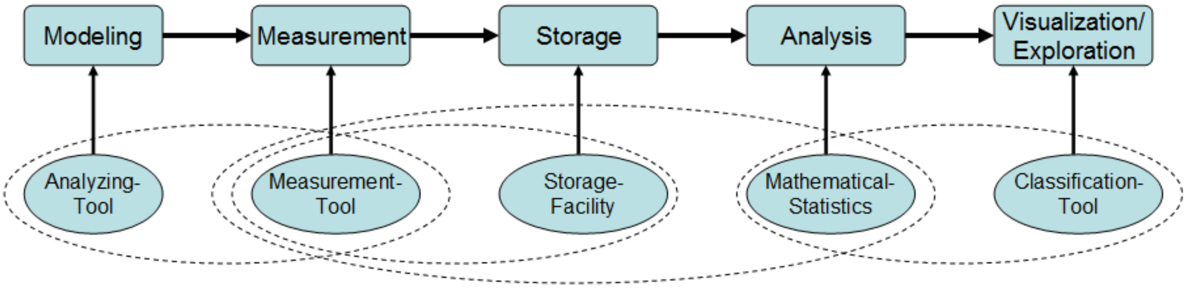


Figure 33: Scope of different CAME tools

According to the different measurement activities CAME tools needs to realize the following features:

- measuring artifacts (source code, structure graphs etc.)
- collecting data

- processing data
- planning and analyzing resources (budget, software resources, human resources)
- generating reports and providing actual analyses

The high importance of measurement tools for software engineering can be summarized with by citing Louis Pasteur with his phrase: “A science is as mature as its measurement tools.”

Nowadays CAME tools can be successful used in all areas of software engineering for instance software product measurement, process quality evaluation, and measurement of resource quality.

The determination into measurement activities and measurement data storage presented in the measurement process frameworks (see chapter 2.3.2. Measurement in Software Development Process) suggest a differential consideration into the thesis in hand, too. At first the situation on measurement tools should be examined and later measurement storage facilities are taken into account. The framework for a novel measurement infrastructure will involve both aspects, of course.

3.1. Software Measurement Tool Situation

Since various different measurement tools exist, the following chapter is intended to present selected representatives for different tool types. Further reading can be found in [Smlab09], [Dumke96] and [EbertDumke07].

3.1.1. Product Quality Measurement Tools

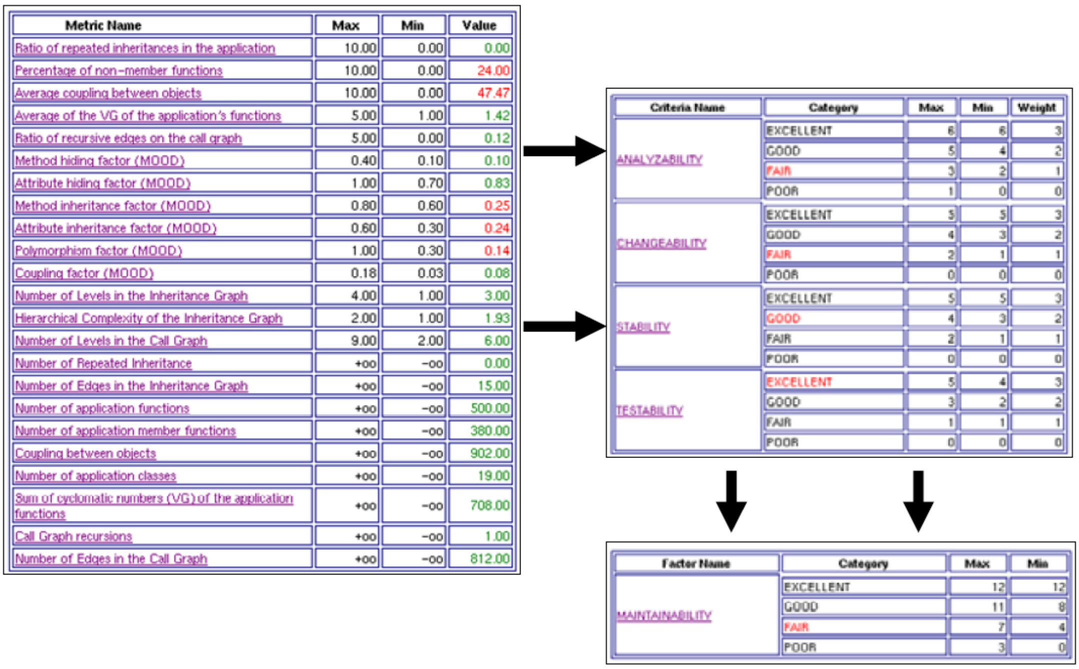
For existing measurement tools in a commercial environment the portfolio of the measurement tool producer Telelogic is identified as a good reference example. Starting in 1993 Telologic became a world leader in software quality assurance tools and was taken over by IBM in 2003. Telelogic produces various tools to support measurement in a broad scope of use cases starting in requirements management (with Telelogic DOORS); across modeling (Telelogic Rhapsody) to software test (Telelogic Tau). For measuring source code and analyzing static and dynamic product quality attributes Telelogics tool Logiscope is a major player in the tool market. By supporting different programming languages, programming paradigms and various types of measures the universal scope of Telelogic Logiscope is comprehensible.

Logiscope is able to calculate measures in different design levels (application, class, function) and therefore analyze different levels of granularity inside a software product.

For the specific quality evaluation of entirely different products the user is able to adapt existing or define a new quality model for a specific use case. The definition of a quality model is done by selecting measures and defining thresholds for each measure. This practice is a well known procedure widely used in the software measurement area [AbrKunz^{+}03].

The measurement results are shown together with the defined quality model thresholds in so-called kiviat diagrams. Some publications appoint this diagram style as radar charts [Lother07].

The derivation of the measurement values is contrariwise to the factor-criteria-metric scheme (see chapter 2.3.3.1. The Goal-Question-Metric Method) All measurement results are used to compute a quality statement for each sub-criteria and the results for the sub-criteria's are used to compute the overall quality statement. This procedure is called the generation of quality statements and shown in Figure 34.

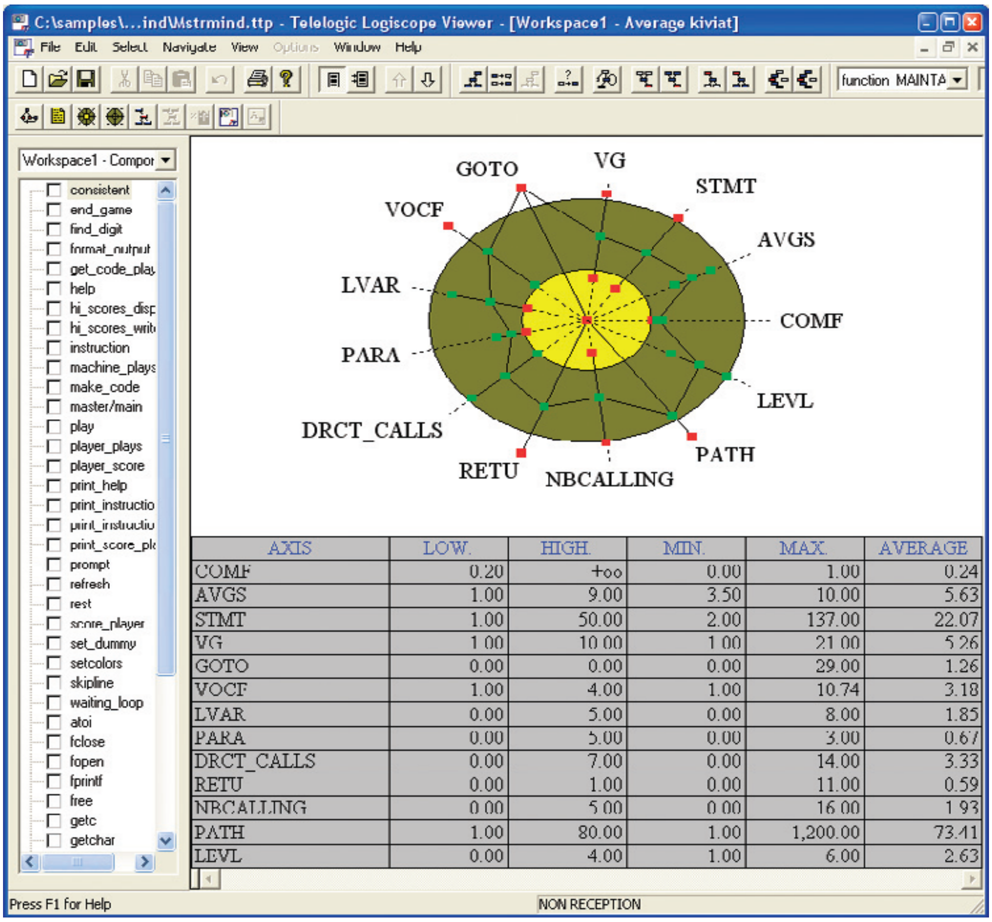


Metric Name	Max	Min	Value
Ratio of repeated inheritances in the application	10.00	0.00	0.00
Percentage of non-member functions	10.00	0.00	24.00
Average coupling between objects	10.00	0.00	47.47
Average of the VG of the application's functions	5.00	1.00	1.42
Ratio of recursive edges on the call graph	5.00	0.00	0.12
Method hiding factor (MOOD)	0.40	0.10	0.10
Attribute hiding factor (MOOD)	1.00	0.70	0.83
Method inheritance factor (MOOD)	0.80	0.60	0.25
Attribute inheritance factor (MOOD)	0.60	0.30	0.24
Polymorphism factor (MOOD)	1.00	0.30	0.14
Coupling factor (MOOD)	0.18	0.03	0.08
Number of Levels in the Inheritance Graph	4.00	1.00	3.00
Hierarchical Complexity of the Inheritance Graph	2.00	1.00	1.93
Number of Levels in the Call Graph	9.00	2.00	6.00
Number of Repeated Inheritance	+oo	-oo	0.00
Number of Edges in the Inheritance Graph	+oo	-oo	15.00
Number of application functions	+oo	-oo	500.00
Number of application member functions	+oo	-oo	380.00
Coupling between objects	+oo	-oo	902.00
Number of application classes	+oo	-oo	19.00
Sum of cyclomatic numbers (VG) of the application functions	+oo	-oo	708.00
Call Graph recursions	+oo	-oo	1.00
Number of Edges in the Call Graph	+oo	-oo	812.00

Criteria Name	Category	Max	Min	Weight
ANALYZABILITY	EXCELLENT	6	6	3
	GOOD	5	4	2
	FAIR	3	2	1
	POOR	1	0	0
CHANGEABILITY	EXCELLENT	5	5	3
	GOOD	4	3	2
	FAIR	2	1	1
	POOR	0	0	0
STABILITY	EXCELLENT	5	5	3
	GOOD	4	3	2
	FAIR	2	1	1
	POOR	0	0	0
TESTABILITY	EXCELLENT	5	4	3
	GOOD	3	2	2
	FAIR	1	1	1
	POOR	0	0	0

Factor Name	Category	Max	Min
MAINTAINABILITY	EXCELLENT	12	12
	GOOD	11	8
	FAIR	7	4
	POOR	3	0

Figure 34: Generation of quality statements [Lother2007]

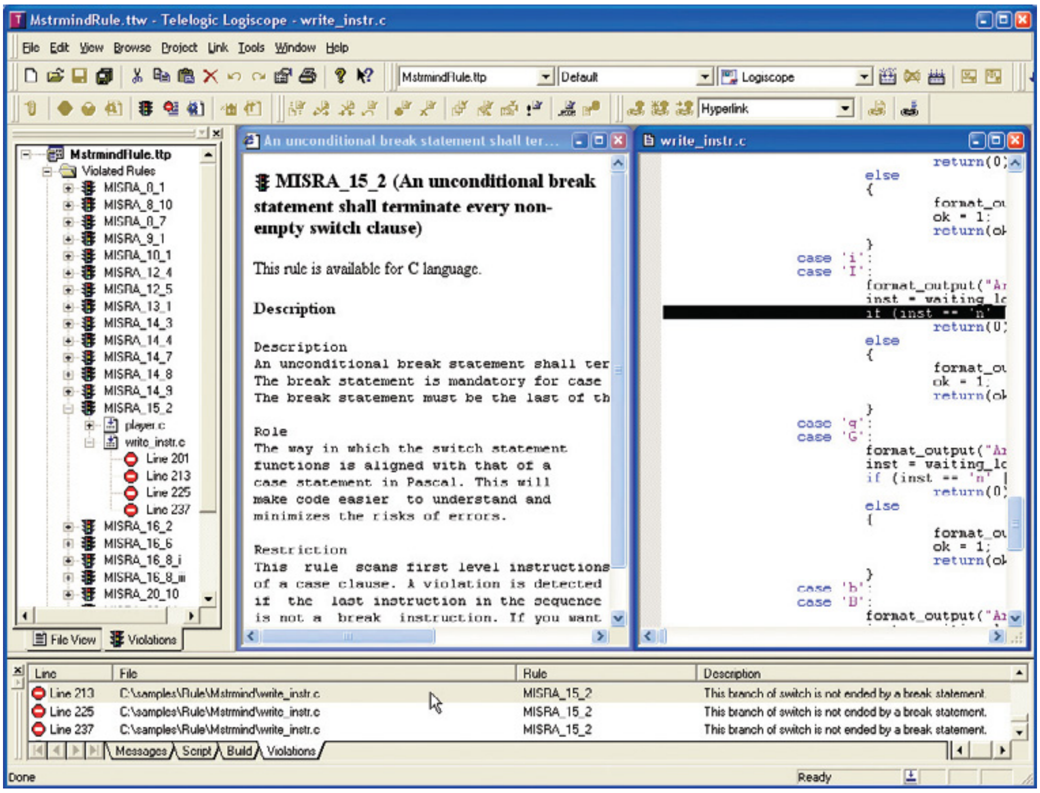


AXIS	LOW.	HIGH.	MIN.	MAX.	AVERAGE
COMF	0.20	+oo	0.00	1.00	0.24
AVGS	1.00	9.00	3.50	10.00	5.63
STMT	1.00	50.00	2.00	137.00	22.07
VG	1.00	10.00	1.00	21.00	5.26
GOTO	0.00	0.00	0.00	29.00	1.26
VOCF	1.00	4.00	1.00	10.74	3.18
LVAR	0.00	5.00	0.00	8.00	1.85
PARA	0.00	5.00	0.00	3.00	0.67
DRCT_CALLS	0.00	7.00	0.00	14.00	3.33
RETU	0.00	1.00	0.00	11.00	0.59
NBCALLING	0.00	5.00	0.00	16.00	1.93
PATH	1.00	80.00	1.00	1,200.00	73.41
LEVL	0.00	4.00	1.00	6.00	2.63

Figure 35: Telelogic Logiscope kiviat diagram

Beside the analysis of the measurement results the user is able to analyze the corresponding source code to identify the reasons and causes for possibly adverse quality evaluation. An example of a kiviat diagram containing measurement values is shown in Figure 35.

The application of the tool is described in a more detailed way in [LotSchm^{+}02] and [LotherBöhm02].



MISRA_15_2 (An unconditional break statement shall terminate every non-empty switch clause)

This rule is available for C language.

Description

```
Description
An unconditional break statement shall ter
The break statement is mandatory for case
The break statement must be the last of th

Role
The way in which the switch statement
functions is aligned with that of a
case statement in Pascal. This will
make code easier  to understand and
minimizes the risks of errors.

Restriction
This  rule  scans first level instructions
of a case clause. A violation is detected
if  the  last instruction in the sequence
is not a  break  instruction. If you want
```

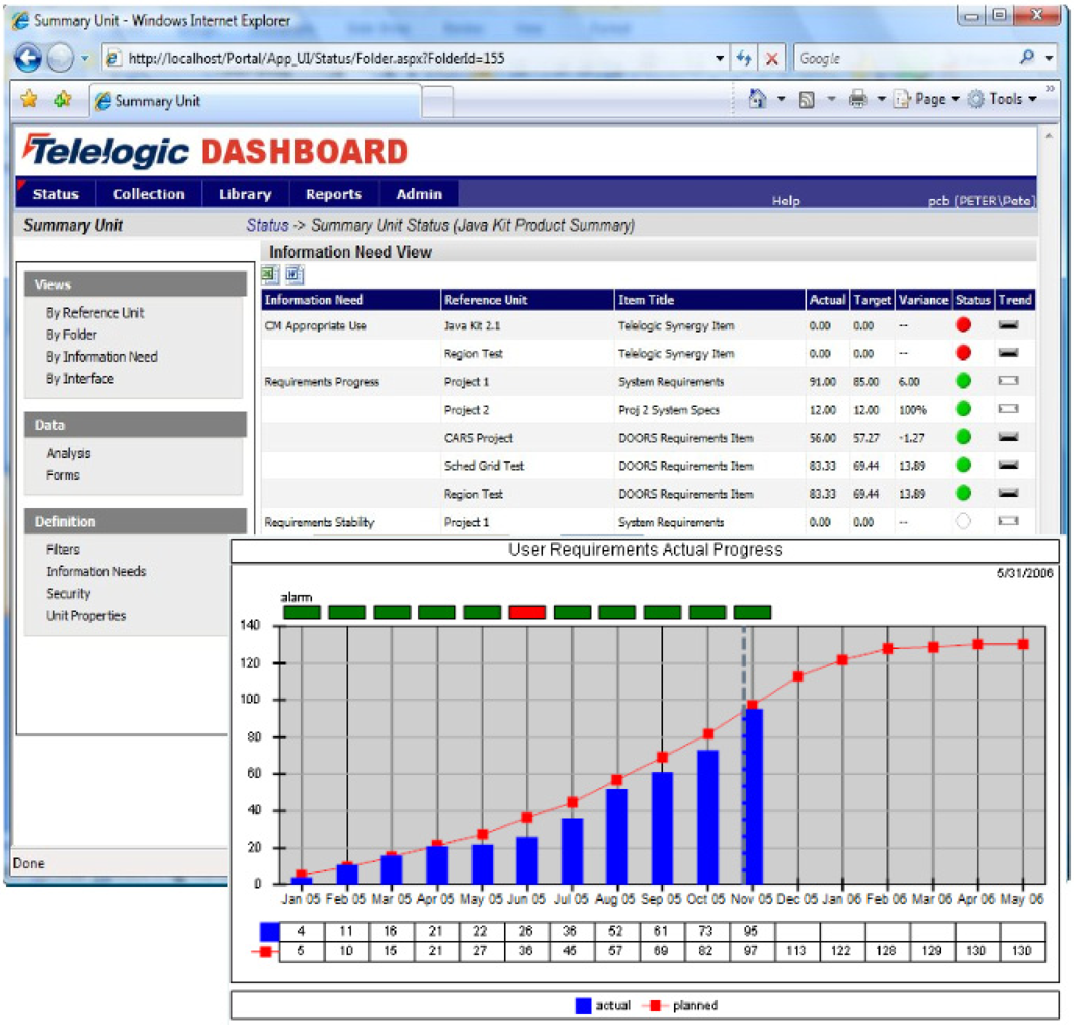
```
                                return(0)
                        else
                        {
                                format_ou
                                ok = 1;
                                return(ok
                        }
                case 'i':
                case 'I':
                        format_output("Ar
                        inst = waiting_lc
                        if (inst == 'n'
                                return(0)
                        else
                        {
                                format_ou
                                ok = 1;
                                return(ok
                        }
                case 'q':
                case 'G':
                        format_output("Ar
                        inst = waiting_lc
                        if (inst == 'n' |
                                return(0)
                        else
                        {
                                format_ou
                                ok = 1;
                                return(ok
                        }
                case 'b':
                case 'B':
                        format_output("Ar
```

Line	File	Rule	Description
Line 213	C:\samples\Rule\Mstrmind\write_instr.c	MISRA_15_2	This branch of switch is not ended by a break statement.
Line 225	C:\samples\Rule\Mstrmind\write_instr.c	MISRA_15_2	This branch of switch is not ended by a break statement.
Line 237	C:\samples\Rule\Mstrmind\write_instr.c	MISRA_15_2	This branch of switch is not ended by a break statement.

Figure 36: Telelogic static source code analysis

In the course of time Telelogic enhances its portfolio with tools for companywide measurement information integration from the different Telelogic measurement tools. The common example is Telelogic Dashboard with the ability to integrate the described Telelogic tools and with various types of visualization capabilities for strategy decision making [Telelogic08]. This intention to create a measurement tool chain is on the one hand supported by the capabilities of integration third party software products like Microsoft Excel and Microsoft Project but on the other hand is limited to specific measurement tools.

As mentioned before the presentation of the Telelogic tool portfolio is only a case in point other tool producers like power software with their tools Essential metrics, Krakatau professional and “visualize it”[powersoftware08] or scientific toolworks with their representative Understand and TrackBack [SciToo08] containing similar functionality and co-operation as described in the case of Telelogic.



Status -> Summary Unit Status (Java Kit Product Summary)

Information Need View

Information Need	Reference Unit	Item Title	Actual	Target	Variance	Status	Trend
CM Appropriate Use	Java Kit 2.1	Telelogic Synergy Item	0.00	0.00	--	🔴	▬
	Region Test	Telelogic Synergy Item	0.00	0.00	--	🔴	▬
Requirements Progress	Project 1	System Requirements	91.00	85.00	6.00	🟢	▭
	Project 2	Proj 2 System Specs	12.00	12.00	100%	🟢	▭
	CARS Project	DOORS Requirements Item	56.00	57.27	-1.27	🟢	▬
	Sched Grid Test	DOORS Requirements Item	83.33	69.44	13.89	🟢	▬
	Region Test	DOORS Requirements Item	83.33	69.44	13.89	🟢	▬
Requirements Stability	Project 1	System Requirements	0.00	0.00	--	○	▭

User Requirements Actual Progress

5/31/2006

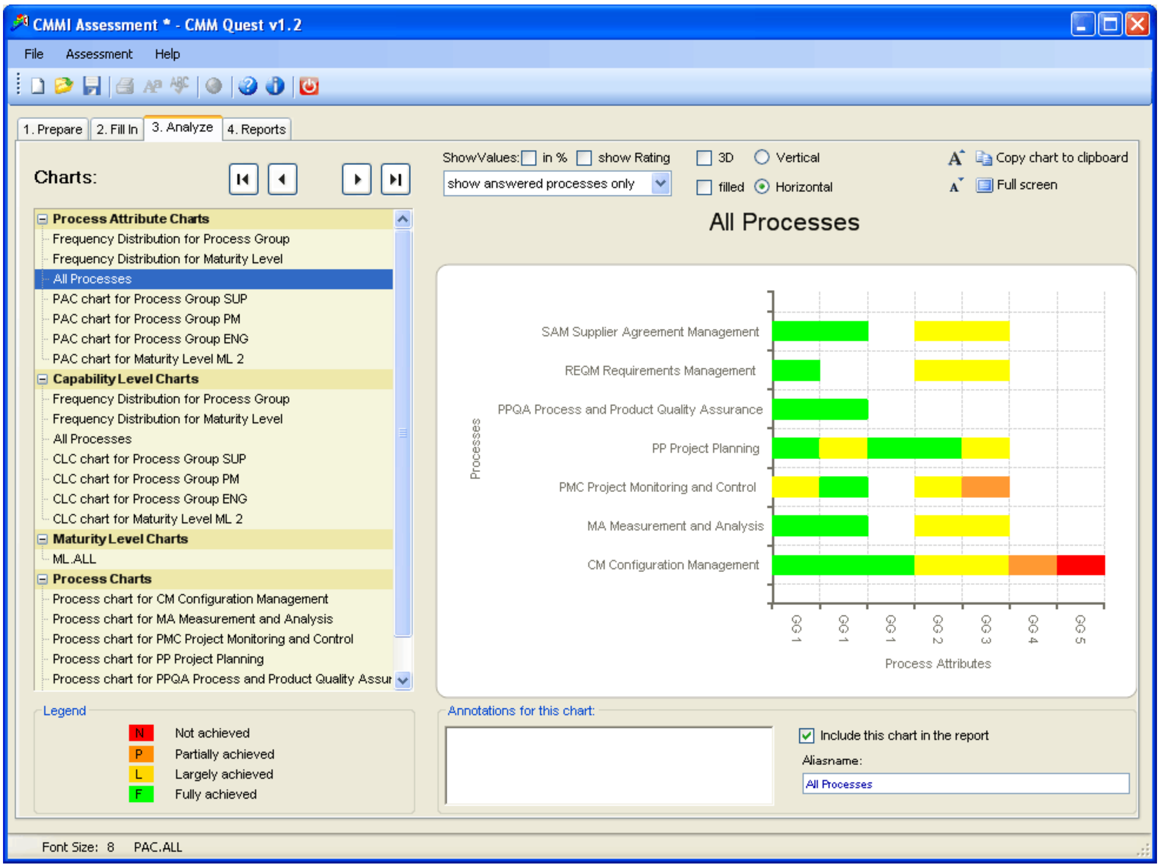
	Jan 05	Feb 05	Mar 05	Apr 05	May 05	Jun 05	Jul 05	Aug 05	Sep 05	Oct 05	Nov 05	Dec 05	Jan 06	Feb 06	Mar 06	Apr 06	May 06
actual	4	11	16	21	22	26	36	52	61	73	95						
planned	5	10	15	21	27	36	45	57	69	82	97	113	122	128	129	130	130

Figure 37: Telelogic Dashboard

3.1.2. Process Quality Measurement Tools

Beside measurement tools arranged alongside the software development lifecycle and creating so-called tool chains for a product quality driven approach, additional measurement tools has been developed to measure the software development process itself.

These process measurement tools are focused on specific process quality standards as there were described in chapter 2.3.2. Measurement in Software Development Process . A common example for suchlike tools is the CMMi assessment tool from CMM Quest [CMMQuest08].


CMMI Assessment * - CMM Quest v1.2
File
Assessment
Help
1. Prepare
2. Fill In
3. Analyze
4. Reports
Charts:
Copy chart to clipboard
Full screen
All Processes
Process Attribute Charts
Frequency Distribution for Process Group
Frequency Distribution for Maturity Level
All Processes
PAC chart for Process Group SUP
PAC chart for Process Group PM
PAC chart for Process Group ENG
PAC chart for Maturity Level ML 2
Capability Level Charts
Frequency Distribution for Process Group
Frequency Distribution for Maturity Level
All Processes
CLC chart for Process Group SUP
CLC chart for Process Group PM
CLC chart for Process Group ENG
CLC chart for Maturity Level ML 2
Maturity Level Charts
ML ALL
Process Charts
Process chart for CM Configuration Management
Process chart for MA Measurement and Analysis
Process chart for PMC Project Monitoring and Control
Process chart for PP Project Planning
Process chart for PPQA Process and Product Quality Assur
SAM Supplier Agreement Management
REQM Requirements Management
PPQA Process and Product Quality Assurance
PP Project Planning
PMC Project Monitoring and Control
MA Measurement and Analysis
CM Configuration Management
Processes
GG 1
GG 1
GG 1
GG 2
GG 3
GG 4
GG 5
Process Attributes
Legend
Annotations for this chart:
Font Size: 8
PAC.ALL

ShowValues: □ in % □ show Rating □ 3D ○ Vertical

show answered processes only □ filled ● Horizontal

N	Not achieved
P	Partially achieved
L	Largely achieved
F	Fully achieved

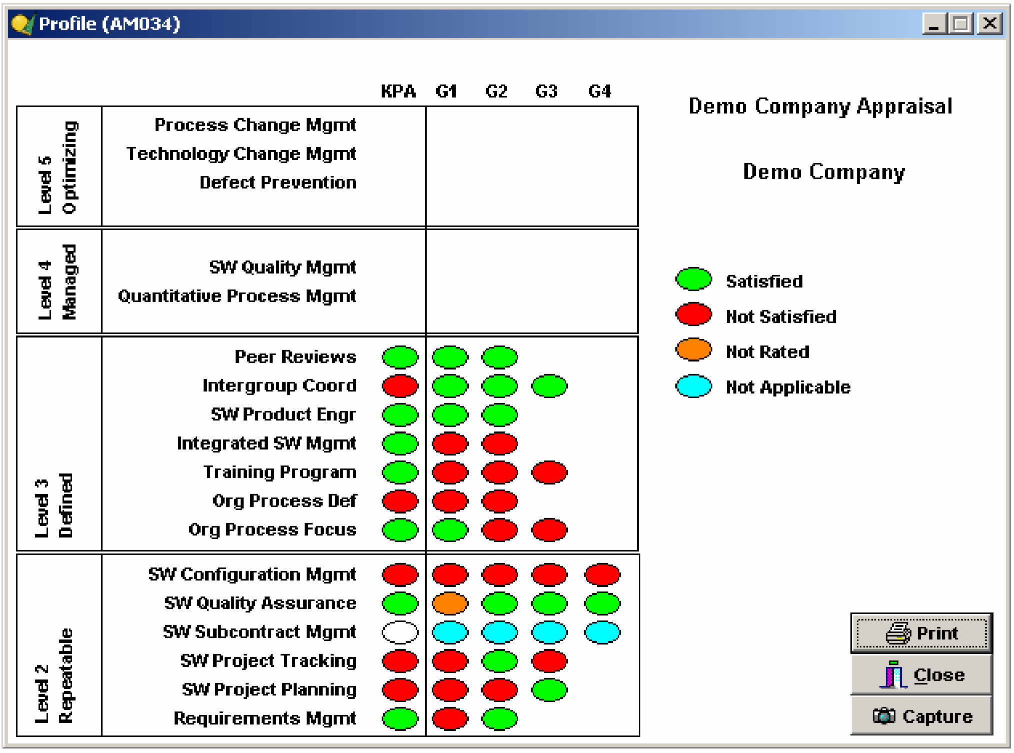
☑ Include this chart in the report

Aliasname: All Processes

Figure 38: CAME tool for CMMI Assessment

The main goal of suchlike tools is to support assessments for process quality evaluation during a certification appraisal. The integration into the development process lacks of the mentioned missing integration or data exchange capabilities of product quality measurement tools and the distrust of development companies to allocate theirs data for quality assessments.

Especially the process alignment with process standards can provide intrinsic value and the standardization of the appraisals can reducing the costs while establish a process improvement cycle. This process improvement can easily be documented through CAME tools for process evaluation.



Demo Company Appraisal

Demo Company

		KPA	G1	G2	G3	G4
Level 5 Optimizing	Process Change Mgmt Technology Change Mgmt Defect Prevention					
Level 4 Managed	SW Quality Mgmt Quantitative Process Mgmt					
Level 3 Defined	Peer Reviews	🟢	🟢	🟢		
	Intergroup Coord	🔴	🟢	🟢	🟢	
	SW Product Engr	🟢	🟢	🟢		
	Integrated SW Mgmt	🟢	🔴	🔴		
	Training Program	🟢	🔴	🔴	🔴	
	Org Process Def	🔴	🔴	🔴		
	Org Process Focus	🟢	🟢	🔴	🔴	
Level 2 Repeatable	SW Configuration Mgmt	🔴	🔴	🔴	🔴	🔴
	SW Quality Assurance	🟢	🟠	🟢	🟢	🟢
	SW Subcontract Mgmt	⚪	🔵	🔵	🔵	🔵
	SW Project Tracking	🔴	🔴	🟢	🔴	
	SW Project Planning	🔴	🔴	🔴	🟢	
	Requirements Mgmt	🟢	🔴	🟢		

🟢 Satisfied
🔴 Not Satisfied
🟠 Not Rated
🔵 Not Applicable

Figure 39: ISD Appraisal Wizard for CMMi Assessments

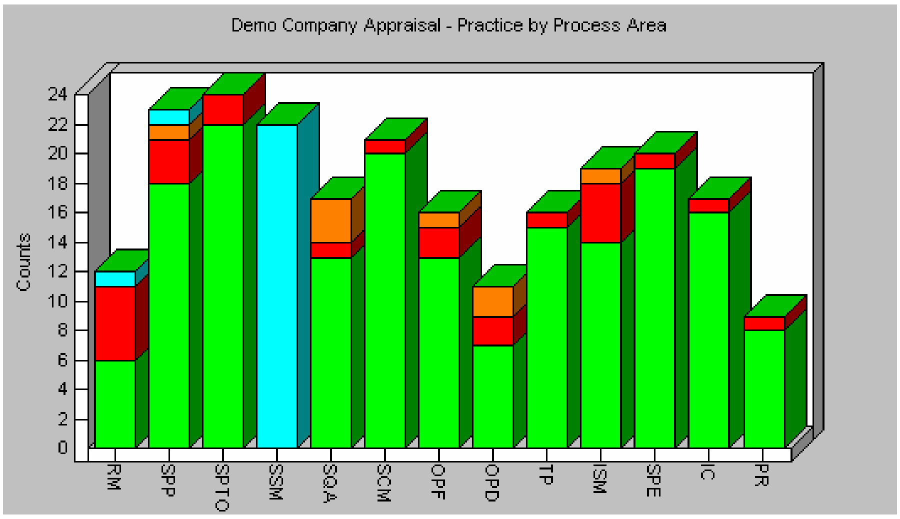


Figure 40: ISD Appraisal Wizard Process Area Viewer

Another major market contender for process measurement and control which should be mentioned is Rational with their core product ProjectConsole [Rational08]. The tool provides information import from other Rational tools and few third-party tools (e.g. Microsoft Project). With this information ProjectConsole offers reporting capabilities to development teams in means of project control and continuous usage of this tool throughout the development process enables analysis in means of process measurement.

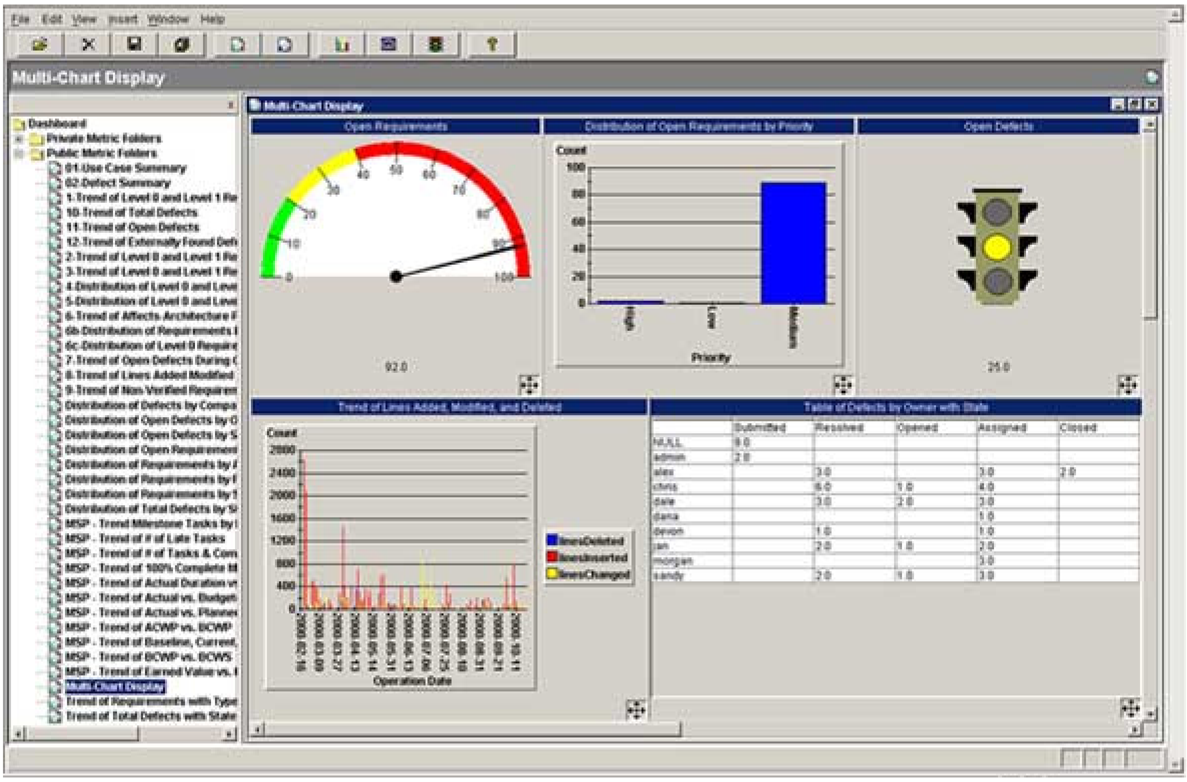

File Edit View Insert Window Help
Multi-Chart Display
Dashboard
Private Metric Folders
Public Metric Folders
01-Use Case Summary
02-Defect Summary
1-Trend of Level 0 and Level 1 Re
10-Trend of Total Defects
11-Trend of Open Defects
12-Trend of Externally Found Def
2-Trend of Level 0 and Level 1 Re
3-Trend of Level 0 and Level 1 Re
4-Distribution of Level 0 and Leve
5-Distribution of Level 0 and Leve
6-Trend of Affects Architecture F
6b-Distribution of Requirements I
6c-Distribution of Level 0 Require
7-Trend of Open Defects During C
8-Trend of Lines Added Modified
9-Trend of Non Verified Requiren
Distribution of Defects by Compo
Distribution of Open Defects by O
Distribution of Open Defects by S
Distribution of Open Requiremen
Distribution of Requirements by A
Distribution of Requirements by I
Distribution of Requirements by T
Distribution of Total Defects by S
MSP - Trend Milestone Tasks by I
MSP - Trend of # of Late Tasks
MSP - Trend of # of Tasks & Com
MSP - Trend of 100% Complete M
MSP - Trend of Actual Duration vs
MSP - Trend of Actual vs. Budget
MSP - Trend of Actual vs. Planne
MSP - Trend of ACWP vs. BCWP
MSP - Trend of Baseline, Current,
MSP - Trend of BCWP vs. BCWS
MSP - Trend of Earned Value vs. I
Multi-Chart Display
Trend of Requirements with Type
Trend of Total Defects with State
Multi-Chart Display
Open Requirements
0 10 20 30 40 50 60 70 80 90 100
92.0
Distribution of Open Requirements by Priority
Count
100 80 60 40 20 0
High
Low
Medium
Priority
Open Defects
25.0
Trend of Lines Added, Modified, and Deleted
Count
2800 2400 2000 1600 1200 800 400 0
2000-02-16 2000-03-09 2000-03-27 2000-04-13 2000-05-14 2000-05-31 2000-06-13 2000-07-06 2000-07-25 2000-08-10 2000-08-31 2000-09-21 2000-10-11
Operation Date
linesDeleted
linesInserted
linesChanged

Table of Defects by Owner with State

	Submitted	Resolved	Opened	Assigned	Closed
NULL	9.0				
admin	2.0				
alex		3.0		3.0	2.0
chris		6.0	1.0	4.0	
dale		3.0	2.0	3.0	
dana				1.0	
devon		1.0		1.0	
jan		2.0	1.0	2.0	
morgan				3.0	
sandy		2.0	1.0	3.0	

Figure 41: Rational ProjectConsole

A detailed analysis about the monolithic characteristics and the capabilities respectively needed efforts for measurement tool integration is presented in chapter *4.3. SOA-capability of Software Measurement Tools* because this research has been done with a focus to a specific integration paradigm called service-oriented architecture (SOA).

3.1.3. General Measurement Tool Capabilities

Existing measurement tools integrate more functionality and tool producers try to establish tool chains within their portfolio to cover as much potential use cases as possible. But obviously interchange from tools of different producers bears a lot of bottlenecks due to few export possibilities and proprietary tool architecture.

To provide an overview about existing tool characteristics the input, output, and automation capabilities are summarized in the following table.

The chosen criteria's are input, output, and automation: Input/output has been chosen because software measurement tools heavily rely on data from various sources (e.g. integrated development environments, UML modeling tools, and work effort databases). The existing of connections to theses applications substantially influences the efficiency of a measurement tool. Secondly the results of software measurement have to be exported for further processing (for example in spreadsheets or other measurement data storage facilities) or reports and graphs have to be created and printed [AuerGraser+03].

The automation of software measurement and thereby the avoiding of expensive manual practices lowers the effort and enhances the accuracy of collected data.

Table 9: Overview about measurement tool capabilities

Tool	**Input**	**Output**	**Automation**
Telelogic Logiscope	Manual Entry, Source Code File	Project Data, CSV, HTML Graphs: PNG	None
Krakatau Professional	Manual Entry, Comma-Separated Values (CSV)	CSV, HTML, PDF, MySQL data file	Generation of quality reports
ISD Appraisal Wizard	Manual Entry, CSV	PDF, PNG; HTML	None
Rational ProjectConsole	Manual Entry, MS Project File	CSV, HTML Graph: PNG	Data Collection, Generation of project website

Summary: Measurement Tool Situation

The consideration of existing measurement tools discovers general short-comings [AuerGraser+03]:

- missing interfaces
- few implemented input file formats
- rare native automation
- customization effort required
- platform dependence
- expensive tool evaluation

As a result of applying suchlike measurement tools in practical environments different drawbacks are extracted from lessons learned [Lother07]:

- measurement tool complexity and lack of standardized interfaces makes it difficult extract measurement data
- expert knowledge is necessary to analyze measurement results
- integration of the measurement tool is necessary to assure a continuous application into a measurement process
- the handling of the tools and the analyses of measurement results is nontrivial
- software developers wanted to have direct access to measured data which is not possible due to lack of interfaces and access possibilities
- no (automated) integration of the measurement process into the development process that leads to lack of support of distributed development processes

One approach to provide the possibility of a tool overlapping analysis and measurement data storage is the establishment of particular measurement storage facilities.

3.2. Software Measurement Repositories

As already mentioned software measurement or more specific software metrics can be used for instance to help estimate project characteristics, measure project progress and performance, or quantify product attributes. But once, having defined a suitable set of metrics data to be collected and analyzed, the question of how to store these data in an appropriate way comes up [BraKunz+05a].

Hence, among others the Canadian *Software Productivity Center Inc. (SPC)* prepared a list of general criteria to meet at best when defining a software measurement data storage facility in line with the establishment of an intra-enterprise software metrics program [BraKunz+05b].

Corresponding to *SPC* (cf. [SPC 04]), this facility should:

- be ***easy to use*** so that users can update and report data with a minimum of trouble.
- be ***flexible*** for the following reasons:
    - so that one can change its structure as the underlying metrics program evolves and new data is collected;
    - so one can perform ad hoc reports and queries on the raw data;
    - and so one can use the data to create metrics other those than identified afore.
- ***interface** to other tools* such as configuration management, defect tracking, and project management systems, ideally. This could greatly simplify data collection and reduce repetition of data within the company.
- be ***large enough*** to contain significant historical information.
- ***avoid** repetition* of data.
- ***provide** the necessary access security*, in case data security is an issue. This includes the security to protect against unauthorized use as well as data corruption.

Of course there is a strong relationship between software measurement data storage facilities and general data storage facilities. Because of that one can identify a common classification in spreadsheet and database management systems within analyzed facilities.

Consecutively both concepts will be described in a nutshell.

3.2.1. Analysis of existing Spreadsheet for Software Measurement

Since there are ambiguous interpretations of the term spreadsheet prevalent in the IT area, generally one has to distinguish between the following two forms in order to be scientifically right.

Spreadsheet program

According to Clermont [Clermont03], a spreadsheet program basically consists of the calculation directive and constant values, which are required to properly specify pertinent formulas.

Spreadsheet (instance)

Also following Clermont [Clermont03], a spreadsheet (instance) can be defined as a spreadsheet program plus the expected input data in the same manner compared to the execution of a program.

Being well aware of the fact that spreadsheet programs are just a piece of software to IT professionals organizing input values in concise columns and rows, for typical spreadsheet program end-users (mostly domain specialists with little or no IT training) a slightly different interpretation approach seems to be advisable:

"For them spreadsheets are a computer supported realization of three tools that are fundamental for our modern civilization: paper, pencil, and calculator." [Clermont 03]

In the context of this publication the abbreviated term spreadsheet is used to pool both meanings in the direction of the instance of a spreadsheet program.

Owing to the release of Microsoft's operating system with a graphical user interface, *Windows*, a new era of spreadsheet systems got introduced with new features, like mouse operation or drag and drop.

Since Microsoft's Excel was one of the first available spreadsheet systems for the new environment, its rivals have not been able to compensate this competitive edge, ever. Consequently, Excel has remained the standard spreadsheet system up to now even implementing a macro-language that enables the user to add arbitrary imperative functions [Clermont03].

Current trends in the development of innovative techniques for spreadsheet systems even include *eXtensible Markup Language* (XML) [Bradley98] representation of data.

In the course of our investigations plenty of spreadsheets of historical software measurement brought together by different organizations or professional associations could be found, whereas only the apparently most established ones shall be illustrated by the following table and examined beneath:

Table 10: Survey and classification of assorted spreadsheets of software measurement

	Architecture Research Facility Dataset	**DACS Productivity Dataset**	**NASA Ames Dataset**	**NASE/SEL Dataset**	**Software Reliability Dataset**	**ISBSG Bench-marking Data CD R10**
Owner:	U.S. Department of Defense (DACS SLED) [15]					ISBSG [16]
Updates:	1979	1989	End of 1970ies	1997	1970ies	Approx. Every 2 Years
Region:	USA	USA	USA	USA	USA	Worldwide
Source:	NRL, U.S. D.o.D.	U.S. Government, Industry	NASA	NASA (GSFC), SEL	NASA, AT&T Bell Labs	International Market Leaders
Environment:	Military	Mixed	Aerospace, Military	Mixed	Mixed	Mixed
Domain:	Problem data	Productivity data	Error data	Complete Software Lifecycle	Failure Interval Data	Project and Productivity Data
Extent of Projects:	1 (192 man weeks) with 142 defects	> 500	33 (3,700 trouble reports)	> 50,000 Records from Status / Runtime Reports	16	2,027
Anonymization:	No	Yes	-/-	-/-	Yes	Yes
Validation:	-/-	-/-	-/-	-/-	-/-	Yes

Metrics:						
LOC	✓	✓	×	-/-	×	✓ (partly)
FP	×	×	×	-/-	×	✓
EFFORT	✓	✓	×	-/-	×	✓
DURATION	×	✓	×	-/-	×	✓
ERRORS	✓	✓	✓	-/-	✓	✓
Others	✓	✓	×	-/-	×	✓

Legend: *✓ = present | × = not present | -/- no information available*

As rendered in Table 10, the market of publicly available but commercially purchasable spreadsheets or as the case may be datasets of software metrics focusing on different domains is basically being dominated by the two organizations DACS (USA) and ISBSG (Australia).

3.2.1.1. DACS SLED

The *Software Lifecycle Empirical/Experience Database* (SLED) which consist of five sub areas has been collected out of diverse sources by the *Data & Analysis Center for Software* (DACS) of the Department of Defense (D.o.D.) of the Government of the United States of America and can be acquired by purchase for a fee of 50 US Dollar. The referring collection process has been initiated by the DACS in order to support the acquisition as well as the maintenance and distribution of empirical data accompanying the software lifecycle (cf. [Dumke00]). Thus, software process quality with regard to research, teaching and commercial concerns can be improved, similarly.

As already stated earlier, therefore five different datasets are being offered that cover all aspects of the software lifecycle:

a) ARF Dataset

In the year 1979 the U.S.-American *Naval Research Laboratories* (NRL), assigned by the U.S. Department of Defense, worked on the development of a computer-based military *Architecture Research Facility* (ARF), while one of its chief developers, D. Weiss, compiled the error data emerging in line with this development endeavor as the *ARF Dataset*. Withal, the carried out software development project had a personnel expenditure of 192 man weeks, whereupon 142 defects stemming from the program code could be discovered [King 04].

A brief survey of the characteristics of the noted error data is shown in Appendix B.

b) DACS Productivity Dataset

Just like almost all other datasets and/or spreadsheets belonging to the DACS SLED the *DACS Productivity Dataset* (DPDS) has been produced by an institution being subordinated to the

U.S. D.o.D. In this special case, the information have been collected by R. Nelson at Rome Research Laboratories of the homonymous Rome Airforce Base on the base of countless, public Government or private industry sources and even got augmented, when he switched over to the DACS organization. This collection of productivity data now comprises information of more than 500 software development projects starting from the 1960ies and quitting in the end of the 1980ies of the past century, stemming from the domains of aviation electronics, aerospace, radar system support, COTS, communication and MIS [King04].

The development aimed at the intention to bring forward and support the analysis of the impact of methodologies for the implementation on the productivity in the area of software development. For this purpose the decision was made to provide valuable software tools for the analysis of the distribution and for the determination of certain relationships between the respective parameters with access to the persisted storage for online users for free. [King04]

For the characterization of a project within the DACS Productivity Dataset beyond a vast quantity of others, primarily ten parameters are applied, which are noted in Appendix B.

c) NASA Ames Error/Fault Dataset

The error and/or defect dataset of the US-American NASA *Ames Aircraft Research Laboratories* (AARL) contains error data had been originated from the end of the 1970ies in line with the development of 33 digital flight control systems on the base of the appraisal of more than 3,700 software trouble reports ([Harrison00] and [King04]).

d) NASA SEL Dataset

In the year 1976 the *Software Engineering Laboratory* (SEL) was founded as an organization funded by the NASA and the affiliated *Goddard Space Flight Center* (GSFC) in order to improve both GSFC software products and processes continuously as well as to positively influence research in the area of measurement and evaluation of the software development process and the effectiveness of methodologies for the software development. In the course of time the SEL achieved an outstanding importance for this scope and could benefit from the collaboration with the *Computer Sciences Corporation of America* (CSC) and the University of Maryland (UMD) by the same token.

After the major update in November 1997 the data collection of the SEL now comprehends more than 50,000 records which contain data of the whole software lifecycle, arranged into five different datasets. Because of the multitude of the consulted sources (mostly component status reports and runtime analyses) a precise subsumption in definite software environments or domains is not feasible, so far [King 04].

The online and at no charge offered software tools for a visualization by scatter plot diagrams and histograms on the base of project statistics as well as scatter plot diagrams and histograms on the base of component statistics are of invaluable virtue.

Unfortunately, an overview of the persisted project parameters is not publicly accessible and can thus not be put on record.

e) NASA Software Reliability Dataset

In the nineteen seventies, J. Musa compiled a set of failure interval data emerging from 16 software development projects of the AT&T Bell Laboratories in order to assist software managers in monitoring test status and in predicting workflows on the one hand and to assist researchers in validating software reliability models on the other hand. Withal, the 16 measured projects were carefully controlled during data collection to ensure a high quality and they originate from different environments; among them is software for real time command and control as well as commercial and military applications [DTIC08].

A detailed description of the dataset can be found in Appendix B

3.2.1.2. ISBSG Benchmarking Data CD Release 10

Cultivating a close cooperation with one of the most important associations for functional size measurement of software called IFPUG the *International Standards and Benchmarking Group* (ISBSG) continuously collects project and productivity data of international and market-leading organizations of the software development sector. Additionally, the ISBSG cares for the anonymity and offers the compilation combined with valuable software tools for a statistical analysis packed on a CD-ROM for purchase; a maximum fee of 850 US Dollar for a single-user license has to be allowed for [ISBSG07].

This collection of product and productivity data has been produced in order to support the emerging of international standards for the software development and to produce a dataset, which is often misattributed a database, as an alternative to commonly very cost-intensive, internal investigations and analyses. Thereby, it mainly aims at resource and effort estimation as well as benchmarking that is of special interest to project managers of the software development. To ensure the quality of the subsequently distributed data, the retrieved information is carefully checked before incorporation. Over and above, with the help of rotational updates in an interval of some two years a high refresh period and steady augmentation of the dataset can be guaranteed: The tenth and latest version of the year 2007 contains records of project and productivity data on over 4000 software development projects emanating from miscellaneous environment of predominantly well established and market-leading organizations from all parts of the world.

A detailed description of the attributes stored in the spreadsheet can be found in Appendix B.

Owing to its straightforwardness, rapidity and hence to the marginal impacts on a limited budget, spreadsheets are exceedingly favored and prevalent for the acquisition of software measurement data. Though they are not suited for a serious conducted data collection in the long-run because the work and management effort on increasing amounts of data cannot be legitimated (cf. [Bernhard 2001]). Furthermore problems due to the lack of structural and systematic consistency pay special attention to a publication of the well-known management consultancy PricewaterhouseCoopers [PriCoo04].

The following table complying with the notes of *SPC* [SPC04] opposes the emerging advantages and disadvantages of spreadsheets and summarizes them, once more:

Table 11: Advantages and disadvantages of spreadsheets

Pro	Contra
✓ Inexpensive and accessible	× No support of ad hoc queries
✓ Easy to configure and use	× Difficult support over network
✓ Built-in graphical reporting facilities	× Not suitable for storing large volumes of data
✓ Easy to update when new data structure is defined	× Difficult to interface to other tools
✓ Familiar to most management	× Little or no security available

3.2.2. Measurement Databases

Consulting technical literature ([HeuerSaake^{+}07], [Vossen00], and [Stein04]), the adjacent definitions can be formulated as:

A database is defined as a structured collection of an exhaustive set of persistent (permanently available) data in an electronic form. These data are acquired under certain aspects and rules, arranged and archived. The management, the access on the data, and its manipulation can be performed with small effort via a database management system.

Software programs for database management (e.g. database access, data manipulation, ensure consistency) form a DBMS as a whole. Thereby they create a software layer between a user and the physical representation of the data.

After all, a database system consists of the subsequently stated constituents:

$$DBS = DB1 \ldots n + DBMS$$

Suchlike DBMS are generally thought to be very efficient in managing tidy and mostly for a long-term use designated amounts of data. They define a database model, which harbors concepts for a uniform characterization and provides operations und descriptive languages for data definition (*Data Definition Language*, DDL), data manipulation (*Data Manipulation Language*, DML) as well as for performing queries (*Query Language*, QL). Furthermore, those systems support transactional concepts with regard to multi-user control and even take care for aspects of data integrity, data security and they also ensure the continuity of the stored data by appropriate arrangements [HeuerSaake^{+}07].

A common summarization of elementary, functional requirements for database management systems that obtained great famousness as the *Nine Rules of Codd* was established in the year 1982 by Codd [Codd82]:

1. Integration (non-redundant data storage by common management)
2. Operation (data definition, storage and manipulation)

3. Catalogue (access to all descriptions of the DB through data dictionary)
4. User views (for different users or applications)
5. Consistency control (assurance of data integrity for events)
6. Access control (avoidance of not-authorized accesses)
7. Transactions (atomic functional units, parallelism, isolation)
8. Synchronization (prevention of mutual influences)
9. Data security (recovery of the stored data)

In the course of time some substantial architecture recommendations and/or standards were set up for DBMS; among them the probably most important ones are: The *Strawman architecture* of CCA/NBS [MatthesSchmidt99], the *Three Level Architecture* of ANSI/SPARC [TsiKlu78] and the *Five Layer Model* of Senko [Senko73] and Härder [Härder87].

In order not to exceed the scope of this thesis the focus shall be limited on the ANSI/SPARC *Three Level Architecture*, which is probably the most wide-spread standard from the conceptual point of view.

The standardization *Three Level Architecture* of ANSI/SPARC has been developed in order to create an independency between data and applications and to support different user views.

Thereunto, a database scheme is sectioned into three consecutive levels, initially [Stein04]:

1. External Scheme (Definition of specific (probably partial) views for different users and/or applications)
2. Conceptual Scheme (Thorough modeling of the entire database by a data model, which is completely independent from a potential system or a concrete implementation)
3. Internal Scheme (Description of the system-specific realization of the database)

Moreover, the architecture of ANSI/SPARC aims at protecting the user of a DBMS against disadvantageous effects in the course of fundamental changes of the system or runtime environment.

This requirement, which is also known under the term *data independency*, can be organized by logical and physical aspects: When the conceptual scheme can be modified without consequences for the external scheme, it is to be the talk of logical data independency, whereas physical data independency can be detected, if the internal scheme can be modified unattached by the conceptual scheme without the appearance of changes in the functionality or even within the applications.

Prior to the concrete characterization of two widespread database models, a definition concerning this matter and stemming from the appropriate literature [HeuerSaake+07] is to be quoted:

„A database model is a system of concepts for the description of databases. Consequently, it defines syntax and semantics of database descriptions, the so-called database schemes."

With regard to this concept formation, a database depicts itself as a tangible peculiarity of a database scheme, which comes in a language that offers assorted and in the database model anchored abstraction concepts.

In line with the examination of classical database models, two main categories can be identified: On the one hand, abstract database models like the *Entity-Relationship* (E-R) model of Chen [Chen76], the modeling with help of the *Object Modeling Technique* (OMT) [RumPre+94] or even the *Unified Modeling Language* (UML) [RumPre+94] exist, which are predominantly suitable for design purposes.

Apart from that, definite database models were developed, which can assist in combination with the results of a preceding design process in implementing a database system.

The two definite database models for the design that are most relevant for the considered practice are now to be brought up: hereunto come within the relational model and several object-oriented models. Besides that, there is a band of additional definite database models like the network or hierarchical model, whose pervasiveness und hence relevance is an only historical one, nowadays ([HeuerSaake+07]).

The relational model

Because of its bribing straightforwardness and accuracy the relational model, which has been inaugurated in 1970 by Codd [Codd70], relishes a high degree in distribution. As its main concepts pertain plain, non-nested tables (sets of tuple), where relationships are established under use of equality of values [HeuerSaake+07].

Object-oriented models

Object-oriented models are usually regarded as an advancement of semantic and nested, relational models: In general (like e.g. the theoretical very profound, object oriented database model of Beeri [Beeri90]), they are endorsed in the structural and operational section as well as in the part of the utilization of higher concepts by object oriented constituents. The structural frame is not only enhanced by types and type constructors, but even by object identities, classes and structural heredity with reference to a class or type hierarchy. Additionally, the object orientation asserts itself in operations for queries and manipulations. Higher concepts like Meta classes, methods, heredity, overriding of methods and encapsulation are used too, of course.

In order to standardize a homogeneous formulation out of the distinctive, object oriented directions of development and database models, the *Object Data Management Group* (ODMG) as an independent constituent of the *Object Management Group* (OMG) created the “de-facto” standard ODMG-97 [ODMG97], which is also known under the term ODMG 2.0.

It articulates an object model for the description of terms and semantic determinations of the related object oriented data model, certain database languages like an *Object Definition Language* (ODL) and an *Object Query Language* (OQL) as an interface for the definition and manipulation of data and/or information. Beyond that, it presents eventualities for a language embedment (bindings) in object-oriented programming languages and identifies parallels to the OMG, CORBA and ANSI-C++ [HeuerSaake+07].

3.2.2.1. Assorted Measurement Databases

In the course of time many different implementations of databases with the aim to storage measurement data could be observed, Table 12 presents an overview about a careful selected set of measurement databases. Some details about each one can be found in Appendix B.

Table 12: Overview about Measurement Databases

	ESA Software Development Database	**FiSMA Laturi Experience Database**	**NASA MDP Data Repository**	**QSM Project Database**	**T-Systems Nova MetricsDB System**	**SPR Knowledge Database**	**PSM Insight Database**	**Ericsson Research Canada MMR**
Cost:	Not Open to the Public, Accessible to Researchers	Mandatory "Experience Pro" Membership (min. 1,000 € p.a.)	Mandatory but Free Online Registration	Not Public, (Indirect) Access as per License Level	Not Public, In-House Use	Not Public, Delivered with SPR Know-ledgePLAN	Free Tool, Mandatory but Free Online Registration	Not Public, In-House Use
Updates:	Permanently	Annually	Permanently	Every 12 to 18 Months	1998	Annually	O.C.	Permanently
Region:	Europe	Europe	Northern America	Worldwide	Europe	Worldwide	O.C.	Canada
Source:	ESA, Industry	Industry	NASA IV&V, Industry	Mixed (Consultancy Clients)	Industry	U.S. Government, Industry	O.C.	Industry
Environ-ment:	Aerospace, Military, Industry	Banking, Trade, Assurances, Administration, Manufactoring	Aerospace, Military, Industry	Mixed	Telecommunications	Management Information Systems	O.C.	Telecommunications
Domain:	Productivity Data	Productivity Data	Error and Product Data	Any Project-related Data	Mutable	Any Project-related Data	Different Templates	Required Product and Process Data for CMMi
Extent of Projects:	157 (Status 1998)	Approx. 600	> 14,000 (plus 2,700 Problem Reports)	Approx. 6,300	-/-	> 8,000	Initially Just 1 Sample Project	-/-
Anony-mization:	Yes	-/-	Yes	No	No	-/-	No	No
Validation:	Yes	Yes	-/-	Yes	No	Yes	O.C.	-/-
Benefit:	Commercial	Commercial	Commercial	Commercial	None, Just a	Commercial	O.C.	Commercial

	and Research	and Research	and Research		Prototype			
Metrics:								
LOC	✓	×	✓	✓	Mutable	-/-	✓	Mutable
FP	×	✓	×	✓	Mutable	-/-	✓	Mutable
EFFORT	✓	✓	×	✓	Mutable	-/-	✓	Mutable
DURATION	✓	×	×	✓	Mutable	-/-	✓	Mutable
ERRORS	×	×	✓	✓	Mutable	-/-	✓	Mutable
Others	✓	✓	✓	✓	Mutable	-/-	✓	Mutable

Legend: *✓= present | × = not present | -/- no information available | O. C. = owner's choice*

The following table complying with the notes of SPC [SPC04] opposes the emerging advantages and disadvantages of (relational) DBMS within this special context and summarizes them, once more:

Table 13: Overview of advantages and disadvantages of software measurement databases

Pro	**Contra**
✓ Effective storage of large volumes of data	× Initial setup is more costly and time consuming
✓ Security and network access	× people may require training to use the database
✓ Support of ad hoc queries and reports	× more expensive to license than spreadsheets
✓ many interface to high quality graphical presentation facilities	
✓ easy to update and customize as data evolves	

3.2.3. General measurement repository characteristics

Especially high acquisition cost and therefore questionable cost/benefit ratios are counterproductive to identified best practices for software measurement. That aggravates the establishment of software measurement programs as stronger as lower the provision of capital is present in distinct software development processes. This situation leads to a very adverse situation especially in small and medium-size enterprises.

But beyond gut instinct scientific research needs empirical analysis to establish new research perspective.

Overall a large diversity of different CAME tools for different measurement approaches and data storage facilities can be observed. Due to the described shortcomings an automation of software measurement could not be observed. The definition of standardized interfaces of measurement data storage facilities enables at least an automation of data collection. The analysis of the state of the art of measurement tools shows that the long standing goal of creating an integrated measurement and management system bearing the capability of synchronizing and coordinating every measurement activity is not reach yet. Keeping in mind that this goal was stated by Knöll nearly twenty years ago [Knöll91].

Nowadays the realization of corporate measurement programs is done by applying different monolithic tools and respectively measurement data management facilities to create a specific tool chain for a specific measurement process. Founded in the existence of more than one measurement process in one development process/company a variety of the measurement tool landscape can be identified.

The general situation in the coverage of measurement approaches by measurement tools can be described as sufficient. The adverse situation regarding the success rate of corporate measurement programs cannot be reasoned with non existence of measurement tools for specific methods or measures.

But nearly all identified process related best practices are considered inadequately.

All monitored measurement programs find application in major companies. The collected empirical data about software measurement is limited to major development processes, too.

The exclusion of small and mid-size companies in software measurement leads to a tendentious situation regarding empirical data which adverse for major companies, too. This one-sided consideration is one reason for the described problems in software measurement processes.

The goal of the thesis in hand is to enhance the success rate and feasibility of corporate software measurement programs by providing a framework for a measurement infrastructure aiming process automation and tool integration.

3.2.4. Review and Evaluation of existing Approaches to tackle the described Drawbacks

Beside the described existing measurement tools and measurement databases some new approaches regarding software measurement and measurement data storage has been published and used in areas unlike software measurement. Even if these approaches are not applied in software engineering industry the capability of being a solution or at least part of the solution should be analyzed in the following chapter. For the ease of reading the order of the topics has been inverted and the chapter starts with new approaches regarding data storage facilities.

3.2.4.1. Measurement Repository Approaches

To overcome the described disadvantages of existing measurement data storage facilities to major approaches not limited to software engineering can be identified: Repository and Experience Factory.

In the described context of IPSE and CASE, Chou (in [Bergin93]) comprising Meta data, modeling and source code repositories, circumscribes the term in that way:

"The repository, also known as an encyclopedia, is a database that stores information about organizational and data structures, strategic goals, functional processes and models, implementation procedures, and related computer programs."

Bernstein, probably being one of the most common researchers in that thematic area of the US-American Microsoft group, defines the term *Repository* in one of his papers [Bernstein97] more generally referring to engineering disciplines:

"A repository is a shared database of information about engineered artifacts, such as software, documents, maps, information systems, and discrete manufactured components and systems (e.g., electronic circuits, airplanes, automobiles, industrial plants)."

Differing only slightly from Bernstein's disambiguation, McClure [McClure92] as a long term repository expert understands the term "repository" in the following sense:

"The Repository is the mechanism for the definition, storage, and management of all information about an organization, its data and its software systems as well as its access."

Considering these characterizations, the aim of a repository's appliance for the software development process becomes apparent: Models and contents of engineered artifacts shall be stored in order to support not only software-based design tools.

According to Bernstein [Bernstein97], in the course of the lifecycle of engineered software artifacts various objects of umpteen diverse types are defined, created, manipulated, and managed by a diversity of tools. They need to share or exchange their data, even though the objects themselves may also be stored in a diversity of storage systems.

3.2.4.1.1. Assorted Repositories with the Potential to Store Measurement Data

Since the repository technology turns out to be a relatively new one bearing the potential to form amazingly all-embracing IPSE and/or CASE environments even having the ability to store software measurement and/or metrics as well as other Meta data accompanying the software development process, many vendors released and still their versions of repositories with more or less prosperity.

Presumably, one of the first (but commercially less successful) attempts to establish such a repository has been undertaken in 1990 as constituent of the product Application Development/Cycle or shortly AD/Cycle of the IBM. When Texas Instruments in cooperation with Microsoft smelled the rat and winded profit in 1994 the development got pressed forward a lot resulting in the Microsoft Repository product line with versions 1.0 in 1997, Version 2.0 of 1998 and the current one 3.0 in 2003 [Microsoft08]. From that time one or as the case may be in parallel other vendors stepped in and presented their repositories, among them parts of the powerful mySAP Business Suite of SAP AG, Unisys Repository (UREP), Rochade of Allen System Group (ASG, initially developed by Viasoft), and even the passed by Universal Directory of Logic Works.

Because it is nearly impossible to elude from Microsoft’s dominance, their repository version is probably the most important and referred one in present. Consequently, in order not to go beyond the scope of the thesis in hand, the explanations should be limited to a short and shallow characterization of the Microsoft Repository and ASG-Rochade.

3.2.4.1.2. Microsoft Repository

In spring 1994 Texas Instruments and Microsoft started a collaborative project to design a Meta data repository to be integrated into the respective company’s tool strategies. Ever since version 2.0 evolved in 1998 when Microsoft Repository 2.0 has been enclosed to MS SQL Server 7 and Visual Studio 6 for the first time, it is now constituent of several succeeding Microsoft products. Starting with Microsoft SQL Server 2000 the repository engine 3.0 is enclosed. Its development aimed at the compatibility to the Microsoft’s object architecture Component Object Model (COM), at simple extensibility as well as version and configuration management.

The architecture, consisting of CASE/CAME tools, an information model, a repository manager, and an underlying database, can be outlined as illustrated in Figure 42 and is described below.

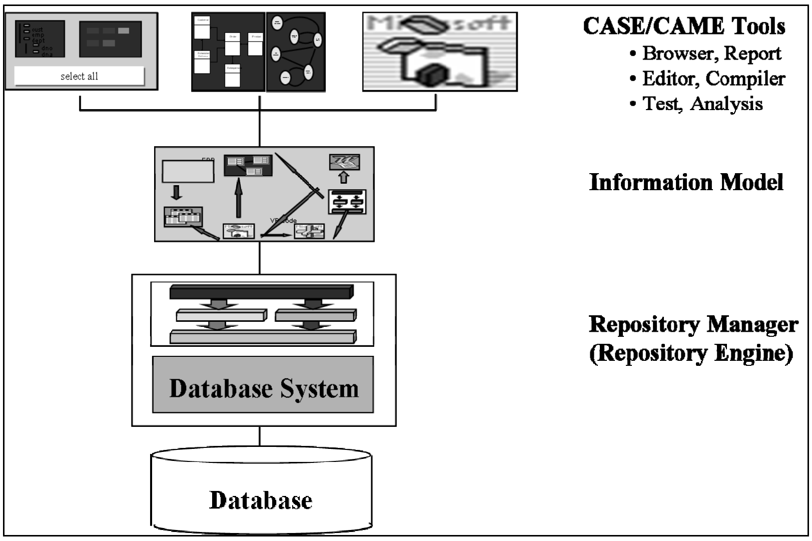


Figure 42: Architecture of Microsoft Repository

Information Model

The contained information model is classed as the Open Information Model (OIM) describing a core model of shareable and reusable type descriptions under use of a set of UML-driven Meta data specifications to circumscribe the structure and the semantics of these Meta data. Targeting the three main areas of Object-Oriented Analysis and Design (OOAD), component description and specification, and database schema and warehousing [McInnis99], its concrete formulations of classes, interfaces, properties, relationships, collections, and models resulting in a so-called Type Information Model (TIM) can either be performed graphically by the Visual Modeler tool or textually by Visual Basic.

Ultimately, these type descriptions are lodged in interface type libraries by standards of the Interface Definition Language (IDL) or custom Type Libraries (TLB). Owing to the pervasiveness of the eXtensible Markup Language (XML) being a text-based format for representing structured data using tags to mark the information, the repository offers special import and export tools, which translate instances of the OIM between the repository and XML. These mechanisms named under the term XML Interchange Format (XIF) can be considered as a standard way of interchanging OIM's instances [McInnis99].

Repository Engine

Preferably layered on a SQL server of Microsoft's product line as underlying storage engine, the repository engine appears as an object management system with Application Programming Interfaces (APIs) resting upon Microsoft's COM and SQL. Of course, the engine features tasks such as relationship and version management, a workspace model and support of repository sessions as claimed by Bernstein ([Bernstein97] and [BerHar$^+$97]).

Additional Software

Besides the standard components of the repository's architecture the Repository Software Development Kit (SDK) and the Modeling Environment are ready for free download. As a modeling and administration environment including OIM, it allows simple creation, enhancement and development of information as well as a check of self-developed information models. Furthermore the SDK has the ability to generate another variant of the repository via Microsoft's pseudo-relational DBMS Access from the OIM. Even an installation script for the OIM, type libraries or header/constant files containing the object IDs for COM can be produced. As a goody, the SDK can even produce relating specification documents in Rich Text Format (RTF).

3.2.4.1.3. ASG-Rochade

The Rochade repository of Allen Systems Group Inc. (ASG) formerly originated by Viasoft is denoted by its software producer as the leading Meta data repository fitting for large and medium-sized organizations.

Believing in the sales promotion documents [ASG08] it provides a "streamlined process for centralizing the management of metadata from sources throughout the enterprise." From a single, centralized, and easily accessible location the repository is designed to store, interrelate, and disseminate information customized to every user's view. Besides supporting reconciliation of business and design models, integration of different CASE tools (e.g. Rational Rose), and resolution of conflicts in data definitions, it also provides versioning, configuration management, and security.

Additionally, the repository brings along outstanding capabilities in data warehousing, component/object management, enterprise architecture, and portfolio management.

Existing applications can be integrated via XML and a wide range of APIs. Information is delivered according to the end-user's preferences or background; even a Web-based access to all stored Meta data is possible. In addition to the standardized supported Common Warehouse Meta-Model (CWM) of OMG, the repository ensures a maximum leverage of Meta data through integrated development facilities resting upon open APIs.

Believing in the notes of McClure [McClure94] the utilization of repositories entails a number of advantages and just one obvious disadvantage, which shall be opposed within the following table:

Table 14: Overview of advantages and disadvantages of repositories

Pro	Contra
✓ Allows shared utilization of system ✓ information far beyond the boundaries ✓ of applications, tools and the system’s ✓ life cycle ✓ Facilitates an integrated multi-user ✓ tool environment ✓ Support of ad hoc queries and reports ✓ Improves the communication between ✓ users and the shared access on ✓ information ✓ Consolidates und removes redundant ✓ information of an organization ✓ Increases system integrity ✓ Relieves system maintenance ✓ Allows the combination of CASE tools of different vendors ✓ Allows the reuse of information beyond the boundaries of the life cycle supporting tools ✓ Alleviates the migration and conversion	✗ Increases management efforts and acquisition costs

Summary: Software Measurement Repository

Summing up pros and cons one has to reason that spreadsheets do not bear the capability to be a driving force for measurement programs. Especially the lack of interfaces to other tools and the lack of security are major drawbacks. And spreadsheets are not able to use nowadays prevalent communications capabilities over ICT (information and communication technology). The situation for measurement databases is less adverse indeed they are more difficult to use and the initial setup is more costly and time consuming but they bear at least the capability to store and mange measurement data over long period in time and provide the necessary functionality to support the analysis of measurement data [BraKunz+05c].

The inadequacies of existing implementations or more related to comprised measurement data rather than database technology. These measurement data is either bounded to a specific measurement method or few specific metrics or the measurement data is bounded to a single measurement field of application inside a single company or department. The strengths of DBMS technologies regarding security and network access promise to deliver sufficient integrations capabilities to use existing measurement databases in new measurement infrastructure solutions [Kunz+06d].

Indeed, the automation of measurement analysis needs adjustments in database technology, too. Possible approaches regarding data description for analysis support are shown in chapter 5.5. Service-oriented Measurement Database

Accessorily, Bernstein [Bernstein97] lists several benefits of a common repository:

1. Since a repository provides storage services, tool developers need not to create tool-specific databases.
2. Furthermore, it forms the fundament, so that tools can work together via easy information sharing without special and cumbersome protocols: By conforming to a data and information model, software tools can share data without knowing the internals of other tools. In contrast to an axiomatic similar data dictionary, a repository can store not only metadata but information about the whole range of object types pertinent to an enterprise.
3. Because a repository is per definition also a database, it is subject to common control services of databases like integrity, concurrency, and access control. Over and above, mechanisms for check out/check in, version and configuration control, notification, context management and workflows exist.

3.3. Measurement Data Integration

An implementation of new techniques in a novel measurement database has to take into account existing databases and it has to support the integration of existing information because of their importance in software measurement processes.

Three major integration approaches have been proposed by the author and should be described next in a nutshell.

A measurement data warehouse is intended to use measurement data integration for data consolidation into a data warehouse by extracting and transforming measurement data from different sources like measurement tools or spreadsheets. The information is stored into measurement data storage by using ETL tools (Extract Transform Load) [Naumann05]. As result the approach contains a central measurement database to serve analysis needs [Wu05]. These analyzing needs can be solved using various analysis possibilities provided by an OLAP server (Online Analytical Processing). The principle architecture is shown in Figure 43.

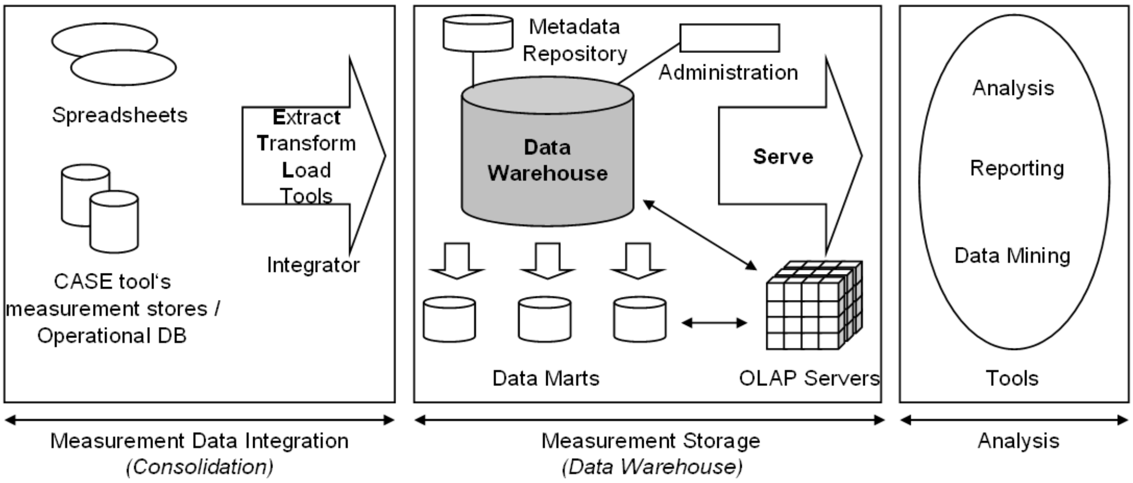

Spreadsheets
CASE tool's measurement stores / Operational DB
Extract Transform Load Tools
Integrator
Metadata Repository
Administration
Data Warehouse
Serve
Data Marts
OLAP Servers
Analysis
Reporting
Data Mining
Tools
Measurement Data Integration (Consolidation)
Measurement Storage (Data Warehouse)
Analysis

Figure 43: Measurement Data Warehouse

Due to one central measurement data storage redundancy can be much reduced. And single analysis functionality bears the capability to integrate multiple measurement data into different views.

A second approach creates a Measurement Data Repository by the use of a mediator. Different wrapper for each data source enables the integration independent from the used technology in the different sources which should be integrated. In this approach the data is kept into the original sources and accessed by the mediator each time an analysis need is demanding the information. From the analysis perspective the case is the same in comparison to the data warehouse approach but from the data storage view each data source has to be maintained and operated. The mediator centered approach is presented in Figure 44.

This approach is a promising one especially if the original data storage is an important source for information in the future and not in the domain of the measurement database owner.

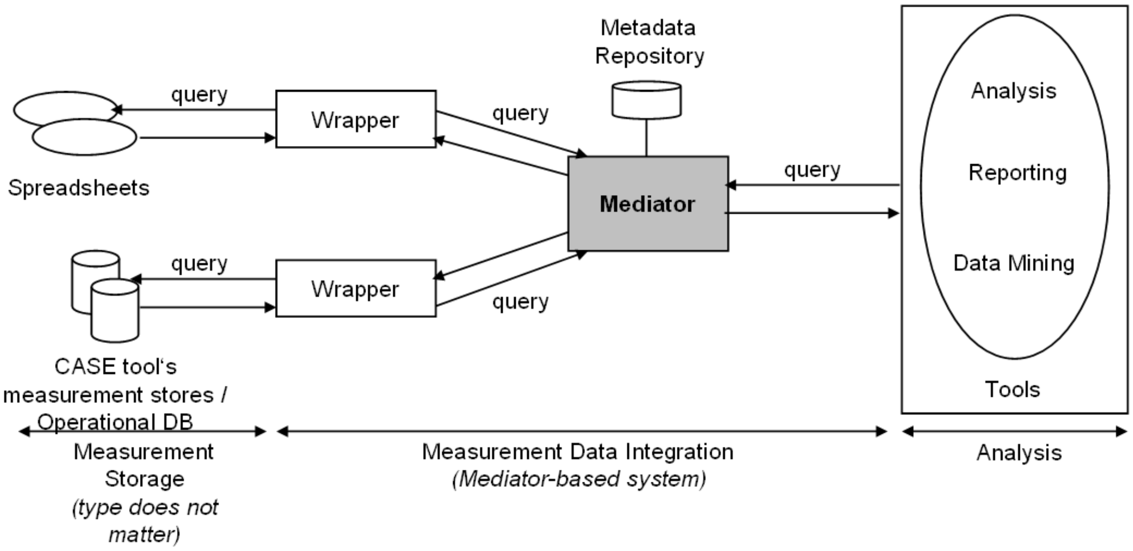

Metadata Repository
query
Wrapper
query
Spreadsheets
Mediator
query
query
Wrapper
query
CASE tool's measurement stores / Operational DB
Analysis
Reporting
Data Mining
Tools
Measurement Storage (type does not matter)
Measurement Data Integration (Mediator-based system)
Analysis

Figure 44: Mediated Measurement Repository

With raising numbers of information systems the amount of data sources raised, too. And this evolution leads to new integration approaches like service-oriented architectures. To uses these techniques in the area of software measurement storage facilities seem to be a promising approach. Especially since the whole measurement infrastructure contains service-oriented paradigms. An alteration at this point would be astonishing and in this case very adverse.

The main architecture is shown in Figure 45. Containing different adapters for measurement data integration and defined interfaces for visualization purposes. The availability of service-interfaces in the different measurement tools or existing measurement facilities is the most important requirement in this approach to support an easy integration of these components.

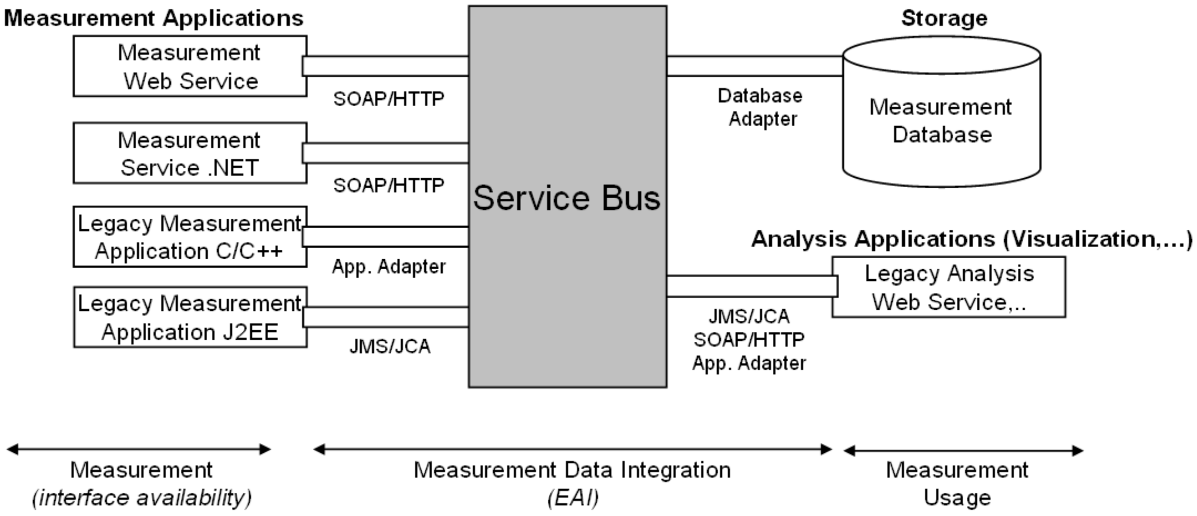

Measurement Applications
Measurement Web Service
SOAP/HTTP
Measurement Service .NET
SOAP/HTTP
Legacy Measurement Application C/C++
App. Adapter
Legacy Measurement Application J2EE
JMS/JCA
Service Bus
Storage
Database Adapter
Measurement Database
Analysis Applications (Visualization,...)
Legacy Analysis Web Service,..
JMS/JCA
SOAP/HTTP
App. Adapter
Measurement (interface availability)
Measurement Data Integration (EAI)
Measurement Usage

Figure 45: Service-oriented measurement database

3.4. Measurement Experience Approaches

Dumke [Dumke03b] defines experience in the environment of Software Engineering as following:

"Experience in the area of software development and utilization is the gained knowledge by practice, case studies, and experiments for the development and utilization of software products comprising the underlying processes and applied resources."

According to the preceding definition, experience can also be seen as immediately "experienced" knowledge resulting from actual software development project business exertion, where it turns out to be a competitive advantage.

Indeed, there are problems in conveying and receiving experiences due to inadequate provision for its validation as well as its origin and suitability for special project contexts, or due to alternating team assemblies for different projects, employee turnover, or organizational expansion. An explicit organizational experience management can help overcoming these shortcomings and improving organizational performance by bringing forward reuse of products, processes and experience stemming from the entire software life cycle.

Aiming at the development of higher quality software systems at lower costs complying with the Quality Improvement Paradigm (QIP) [Basili85], this challenge leads to the development of so called Experience Factories incorporating repositories, which Basili defines as following:

"The Experience Factory is the organization that supports reuse of experience and collective learning by developing, updating and delivering upon request to the project organizations clusters of competencies [...] as experience packages." [BasCal$^+$92].

"The Experience Factory is a logical and/or physical organization that supports project developments by analyzing and synthesizing all kinds of experience, acting as a repository for such experience, and supplying that experience to various projects on demand." [BasCal$^+$94].

Both disambiguations suggest a philosophy of logical separation (see Figure 46 for details) between project development and systematic, organizational learning, and packaging of reusable experiences in order to avoid neglecting one of the tasks, sooner or later.

While the Project Organization has to meet a development project's constraints like duration and budget, business units and/or organizations are striving for the improvement of their products and development processes with the aid of reuse and error prevention via experience packaging by an Experience Factory (EF).

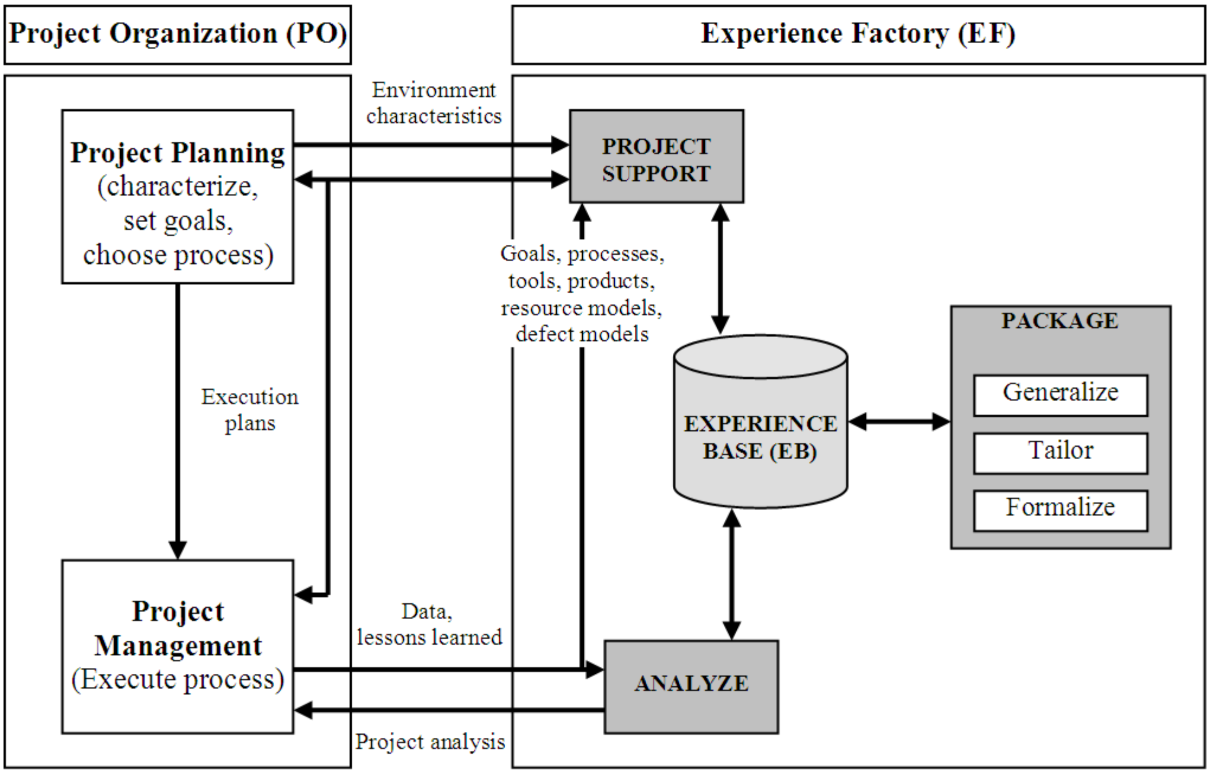

Project Organization (PO)
Experience Factory (EF)
Project Planning (characterize, set goals, choose process)
Environment characteristics
PROJECT SUPPORT
Goals, processes, tools, products, resource models, defect models
Execution plans
EXPERIENCE BASE (EB)
PACKAGE
Generalize
Tailor
Formalize
Project Management (Execute process)
Data, lessons learned
ANALYZE
Project analysis

Figure 46: The concept of Basil's Experience Factory

As mentioned before, the Experience Factory approach is based on the Quality Improvement Paradigm (QIP), which consists of six steps performed within projects and two feedback cycles [Basili85]: 1) characterize project goals, 2) set goals, 3) choose a process, 4) execute, 5) analyze results, and 6) package the results of analysis.

Supplementary, two feedback cycles are defined: a project feedback cycle (feedback provided during the execution phase of the project) and an organizational feedback cycle (feedback provided to the project execution and at the completion of the project). The definition of the QIP also implies that organizational and product improvement requires continually accumulating knowledge and storing that knowledge in a form that can be easily accessed, used, and modified.

So the first three steps of the QIP in terms of project planning (namely characterize, set goals, choose process) are performed by the Project Organization. The results are transmitted as specifications and/or environment characteristics to the Experience Factory or more detailed to the project support.

By virtue of the received data, the Experience Factory tries to find adequate and matching artifacts within the Experience Base. These artifacts can be of different kind (goals, processes, tools, and so on). In case no adequate artifacts can be found, the impact and attention of the project is raised, since there are still no comparable projects available.

In the fourth phase of the QIP the project is executed (execute process) by the project organization or more specific project management. As a result the product to be created and experiences (data, lessons learned) as well as knowledge emerge, which are forwarded to the Experience Factory. Having obtained the new information, the analysis division of the Experience Factory quits the QIP by analyzing and packaging the gained experiences. The Experience Base is enhanced and improved by the new data in order to sharpen it to be more effective and suitable for following projects.

Summary: Measurement Experience Bases

In conclusion both approaches can be described as an enhancement of data with additional information. This information is meta-information which makes it easier to use the original stored date. This so called semantic data is a major driving force in many fields of application [Mencke08]

Since their target is to collect and store date rather than cover the complete software measurement process the techniques has to be integrated in holistic measurement approaches to benefit from the advantages in comparison to existing measurement data storage approaches.

3.5. Software e-Measurement

To provide next generation measurement tools three major approaches can be identified: the usage of ICT in e-Measurement, the creation of measurement sensor networks in an area so called software telemetry and the usage of software agents in the area of agent-oriented software measurement.

The usage of information and communication technology (ICT) in many fields of application in IT industry has changed the way developing software. To tackle this challenge the question arises how web technologies can support measurement processes [LotBraKunz+05].

Mainly three characteristics from web technologies are transformed to software measurement [Lother07]:

- ubiquity (in reach from everywhere, applied when desired)
- pervasiveness (wide-spread, involvement of different communities)
- mobility (usable independent of user’s position)

Thus, the term e-Measurement can be defined as: „the process of the quantification of objects or component’s attributes according to selected measurement goals by using the capabilities of ICT technologies.” [Lother07] [ErnstHage+02]

The structure of e-Measurement approaches and the interplay with the IT area are shown in Figure 47.

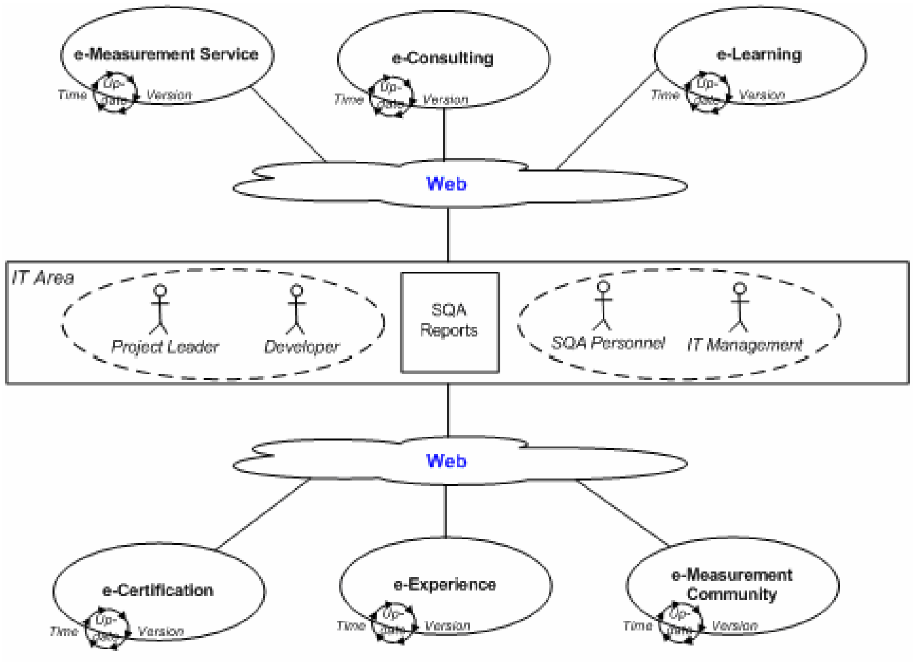


Figure 47: e-Measurement approach for Software Quality Assurance [Lother07]

The e-Measurement approach is characterized by the different elements which are connected by the internet. Each component realizes a specific functionality to enable e-Measurement based software quality assurance [Lother07].

The **e-Measurement Service** can be described as

$$MP_{e-Service}^{e-Measurement}:$$

$$(G \times A \times M)^{T_{Web\_technology}, P} \rightarrow (Q \times E)^{T_{Web\_technology}, P}$$

$$\rightarrow (V \times U)^{T_{Web\_technology}, P} \rightarrow E' \times A'$$

with a simple explanation as *e-Service* ∈ {*global_production, outsourced*} and *Web_technology* ∈ {*document-based, dynamic, semantic, service, mobile, agent, operational*} and the indexes characterizing the main kinds of Web technology.

The **e-Measurement Community** as a virtual environment for the measurement community including features for knowledge transfer, communication, cooperation, and coordination activities is characterized by

$$MP_{e-Community}^{e-Measurement}: (G \times A \times M)^{T_{Web\_technology}, P_{system\_operationality}}$$

$$\rightarrow (Q \times E)^{T_{Web\_technology}, P_{system\_operationality}}$$

$$\rightarrow (V \times U)^{T_{Web\_technology}}\ {}^{P_{system\_operationa\ lity}} \rightarrow E' \times A'$$

with the same kind of description as *e-Community* ∈ {*P2P, research team, cooperating team, organization, competence network*}, *Web_technology* ∈ {*document-based, dynamic, semantic, service, mobile, agent, operational*} and *system_operationality* ∈ {*coordination, conferencing, cooperation, collaboration*}.

Essential backgrounds as **e-Repository** and/or **e-Experience** can be described in a simplified manner as

$$MP^{e-Measurement}_{e-Experience} : (G \times A \times M)^{T_{Web\_technology}}{}_{,}{}^{P_{system\_operationa\ lity}})$$

$$\rightarrow (Q^{Web\_technology} \times E^{Web\_techno\ logy})^{T_{Web\_technology}}{}_{,}{}^{P_{system\_operationa\ lity}}$$

$$\rightarrow (V^{Web\_techno\ logy} \times U^{Web\_technology})^{T_{Web\_techno\ logy}}\ {}^{P_{system\_operationa\ lity}}$$

$$\rightarrow E'^{Web\_technology} \times A'$$

whereas *e-Experience* ∈ {*information basis, repository, knowledge data basis, experience factory*}, *Web_technology* ∈ {*document-based, dynamic, semantic, service, mobile, agent, operational*} and *system_operationality* ∈ {*coordination, conferencing, cooperation, collaboration*} [Lother07].

The **e-Quality Service** are helpful Web-based activities and are described as

$$MP^{e-Measurement}_{e-Quality} : (G \times A \times M^{Web\_technology})^{T_{Web\_technology}}{}_{,}{}^{P_{system\_operationality}}$$

$$\rightarrow (Q^{Web\_technology} \times E^{Web\_technology})^{T_{Web\_technology}}{}_{,}{}^{P_{system\_operationality}}$$

$$\rightarrow (V^{Web\_technology} \times U^{Web\_technology})^{T_{Web\_technology}}\ {}^{P_{system\_operationality}}$$

$$\rightarrow E'^{Web\_technology} \times A'$$

with an explanation as *e-Quality* ∈ {*information, certification, consulting, estimation*}, *Web_technology* ∈ {*document-based, dynamic, semantic, service, mobile, agent, operational*} and *system_operationality* ∈ {*coordination, conferencing, cooperation, collaboration*}.

Especially, the **e-Control** summarizes a lot of Web technologies and methodologies in order to perform this operational kind of Web systems, described as

$$MP_{e-Control}^{e-Measurement}:$$

$$(G \times A^{Web\_technology} \times M^{type\_of\_measurement})^{T_{Web\_technology}}, {}^{P_{system\_operationality}}$$

$$\rightarrow (Q^{Web\_technology} \times E^{Web\_technology})^{T_{Web\_technology}}, {}^{P_{system\_operationality}}$$

$$\rightarrow (V^{Web\_technology} \times U^{Web\_technology})^{T_{Web\_technology}}, {}^{P_{system\_operationality}}$$

$$\rightarrow E'^{Web\_technology} \times A'$$

with the details as *e-Control* ∈ {*evaluation, improvement, managing, controlling*}, *Web_technology* ∈ {*document-based, dynamic, semantic, service, mobile, agent, operational*}, *system_operationality* ∈ {*coordination, conferencing, cooperation, collaboration*} and *type_of_measurement* ∈ {*modeling*, *measurement, evaluation, application*} [Kunz$^+$05].

Finally, the **Measurement e-Learning** [Kunz$^+$05] as one of the measurement training aspects can be formalized as

$$MP_{e-Learning}^{e-Measurement}:$$

$$(G^{Web\_technology} \times A^{Web\_technology} \times M^{Web^{measurement\_operation}})^{T_{Web\_technology}}, {}^{P_{system\_operationality}}$$

$$\rightarrow (Q^{Web\_technology} \times E^{Web\_technology})^{T_{Web\_technology}}, {}^{P_{system\_operationality}}$$

$$\rightarrow (V^{Web\_technology} \times U^{Web\_technology})^{T_{Web\_technology}}, {}^{P_{system\_operationality}}$$

$$\rightarrow E'^{Web\_technology} \times A'^{Web\_technology}$$

whereas it holds that *e-Learning* ∈ {*learning, repetition, consultation, practice, examination*}, *Web_technology* ∈ {*document-based, dynamic, semantic, service, mobile, agent, operational*}, *system_operationailty* ∈ {*coordination, conferencing, cooperation, collaboration, consulting*} and *measurement_operation* ∈ {*artefactBasedOp, quantificationBasedOp, valueBasedOp, experienceBasedOp*}.

The main benefit of e-Measurement leads to the availability of such e-Services and e-Supports. Therefore, the measurement level could be characterized as *immediate level* mainly. Otherwise, using e-Measurement the case of outsourced measurement is the typical one. A usual measurement level description of measurement e-Services as *external process evaluation* could be given as following.

$$MP^{\text{aspect\_ori ented}}_{\text{outsourced}} : (G^{\text{evaluation}}_{\text{internal\_goals}} \times A^{\text{process}}_{\text{original}}$$

$$\times M^{\text{assessment}}_{\text{estimation}})_{T}{}^{\text{semi\_automatic}}_{\text{one\_meas.\_phases}}{}_{,P}{}^{\text{practitioner}}_{\text{meas.\_expert\_staff}}$$

$$\rightarrow (Q^{\text{ordinal scale}}_{\text{single\_value}} \times E^{\text{analogy}}_{\text{threshold}})$$

Another example of e-Measurement based on the “Web-based Measurement” at the SML@b (http://www.smlab.de) as *Java measurement service* has the following measurement characteristics (again as immediate measurement level) [Kunz+05].

$$MP^{\text{aspect\_oriented}}_{\text{outsourced}} : \quad (G^{\text{evaluation}}_{\text{internal\_g oals}} \times A^{\text{product\_aspects}}_{\text{original}}$$

$$\times M^{\text{assessment}}_{\text{measuremen t}})_{T}{}^{\text{semi\_automatic}}_{\text{some\_meas.\_phases}}{}_{,P}{}^{\text{researcher}}_{\text{meas.\_expe rt\_staff}}$$

$$\rightarrow \quad (Q^{\text{ordinal scale}}_{\text{single\_val ue}} \times E^{\text{intuition}}_{\text{threshold}})$$

The best case of measurement level in e-Measurement could be a *remote service of e-Control* (as server management) including the following measurement characteristics.

$$MP^{\text{capability\_oriented}}_{\text{outsourced}} :$$

$$(G^{\text{managing}}_{\text{goals\_in\_u se}} \times A^{\text{resources}}_{\text{original}} \times M^{\text{controlling}}_{\text{measurement}})_{T}{}^{\text{automatic}}_{\text{whole\_meas urement}}{}_{,P}{}^{\text{practitioner}}_{\text{meas.\_appl\_staff}}$$

$$\rightarrow (V^{\text{ratio\_scale}}_{\text{cockpit}} \times U^{\text{software\_u nit}}_{\text{standardiz ed}})_{T}{}^{\text{automatic}}_{\text{whole\_measurement}}{}_{,P}{}^{\text{practitioner}}_{\text{meas.\_appl\_staff}}$$

$$\rightarrow E^{\text{formula}}_{\text{extension}} \times A^{\text{resources}}_{\text{controlled}}$$

Otherwise, simple relationships could be built comparing the traditional kinds of measurement described in the section before. It is simple to see that holds

$$MP^{traditional}_{first\_metrics\_appl.} \prec MP^{e-Measuremen\ t}_{e-Service} \prec MP^{traditiona\ l}_{product\_qu\ ality\_assu\ rance.}$$

and

$$MP^{e-Measuremen\ t}_{e-Control} \prec MP^{traditional}_{project\_controlling}$$

where the non obvious improvements of e-Measurement is reasoned in their better kind of availability and more (world-wide) involved experiences as described above.

Summary: e-Measurement

In general e-Measurement is able to extend the availability and accessibility of measurement tools and enables centralized software management in distributed software development [LotDumBraKunz+05]. A comprehensive description and a very detailed presentation of the software e-Measurement approach can be found in [Lother07].

Especially the accessibility of measurement results by the provision of reports using various formats and communication ways of the internet has been addressed successfully by measurement tool producers. A lot of the described e-Measurement aspects have found their way to practical implementations in large measurement tool suites. Unfortunately the usage of ICT in the different implementation of tool manufactures was not following a specific guideline or industry standard. Only general standards or file formats has been used [DumBraKunz+05]. As the result the capabilities of the software measurement tool enhanced only in perspective of single tool portfolio. The interplay of tools from different tool manufactures is still challenging [AuerGraser+03].

Other disadvantages of existing tools (automation, tailored functionality) are not aimed by this approach [DumkeKunz+05b].

3.6. Agent-oriented Software Measurement

Software agents can be applied to solve new types of problems such as *dynamic open systems:* the structure of the system itself is capable of changing dynamically and its components are not known in advance, can change over time, and may be highly heterogeneous. Usually, the AOSE would be divided in the three areas of *software agent, multi-agent systems (MAS)* and *MAS development* (see [BauerMüller04], [CiaWoo01], [KnapikJohnson98], and [PanaitLuke06]).

Software agents: The essential components of a software agent form a measurement point of view in the following scheme [Wille05].

Multi-agent systems: The viewpoints of agent-based systems – especially multi-agent systems (MAS) - are generally defined in architecture models. We will also start with a general description of the MAS aspects as shown in the following figure [DumkeKunz+05a].

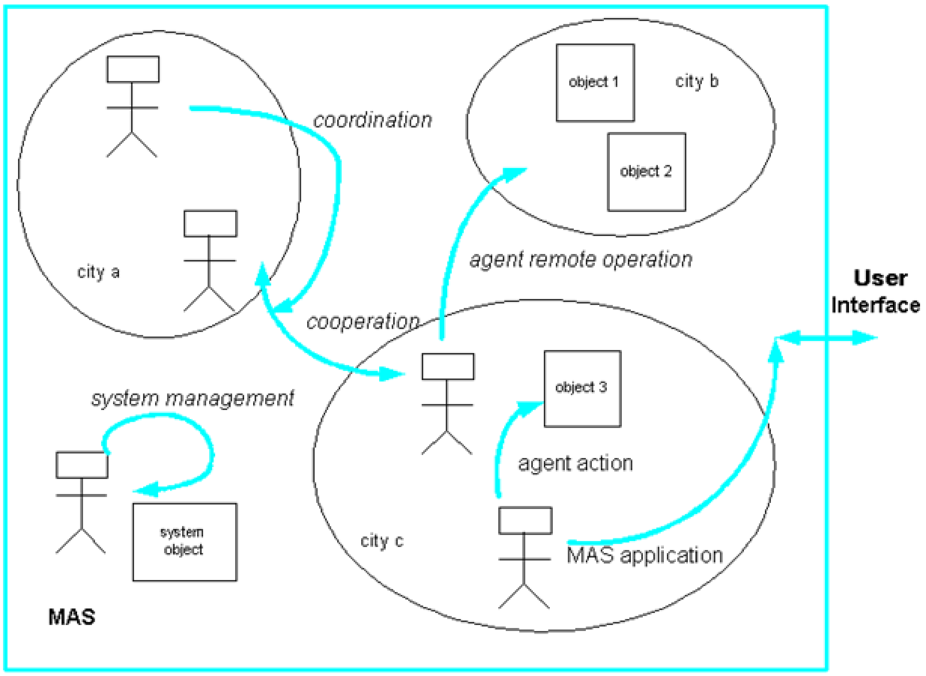


Figure 48: General components of multi-agent systems (MAS)

MAS development: The specification, design and implementation of agent-based system and/or MAS differ from the OO development by starting with subjects (roles) and introducing a training phase after the system implementation. The following figure shows this AOSE development phase involving measurement and evaluation characteristics [MenckeDumke07].

First, the description of measurement of software agents considering the new kind of controlling by the agents themselves is described as:

$$MP^{\overset{AOSE}{agents}}:$$

$$(G^{managing}_{agent\_intention} \times A^{agents}_{original} \times M^{controlling}_{measurement})_{T}{}^{automatic}_{some\_measurement}$$

$$\rightarrow (V^{ratio\_scale}_{repository} \times U^{software\_unit}_{quasi\_standard})_{T}{}^{automatic}_{some\_measurement}$$

$$\rightarrow E^{agent\_knowl\ edge\_basis}_{extension} \times A^{improved\_agent}_{controlled}$$

Especially, the measurement methods μ_{agent} can be summarized as

$$measurement \in \{ \mu^{\overset{size}{agent}}, \mu^{\overset{structure}{agent}}, \mu^{\overset{complexity}{agent}}, \mu^{\overset{functionality}{agent}},$$

$$\mu^{\overset{description(development)}{agent}}, \mu^{\overset{description(application)}{agent}}, \mu^{\overset{description(publication)}{agent}},$$

$$\mu_{agent}^{communicaton}, \mu_{agent}^{interaction}, \mu_{agent}^{learning}, \mu_{agent}^{adaptation}, \mu_{agent}^{negotiation},$$

$$\mu_{agent}^{collaboration}, \mu_{agent}^{coordination}, \mu_{agent}^{cooperation}, \mu_{agent}^{reproduction},$$

$$\mu_{agent}^{performance}, \mu_{agent}^{specialization}\}$$

Usually, agent measurement means controlling considering some of the product characteristics during the run time. This situation can also be established for the multi-agent system itself. Furthermore, agent controlling does not include any personal resources explicitly. Hence, the high level of software measurement for agent technology is described as:

$$MP_{inhouse}^{capability\_oriented} :$$

$$(G_{goals\_in\_u\ se}^{managing} \times A_{original}^{product} \times M_{measurement}^{controlling})_{T_{some\_measu\ rement}^{automatic}}$$

$$\rightarrow (V_{repository}^{ratio\_scale} \times U_{quasi\_standard}^{software\_unit})_{T_{some\_measurement}^{automatic}}$$

$$\rightarrow E_{extension}^{formula} \times A_{controlled}^{product}$$

Otherwise, the process of agent and MAS development could be classified as an immediate measurement level. The following description demonstrates this case of software measurement ingredients.

$$MP_{inhouse}^{aspect\_oriented} :$$

$$(G_{external\_goals}^{managing} \times A_{original}^{process}$$

$$\times M_{measurement}^{improvement})_{T_{some\_measurement}^{semi\_automatic}, P_{measurement\_application\_staff}^{practitioner}}$$

$$\rightarrow (Q_{repository}^{ordinal\_scale} \times E_{threshold}^{criteria})_{T_{some\_measu\ rement}^{semi\_autom\ atic}, P_{measurement\_application\_staff}^{practitioner}}$$

$$\rightarrow E_{extension}^{rules\_of\_thumb} \times A_{improved}^{process}$$

Summary: Agent-oriented Software Measurement

The agent-based measurement level comparing to the other paradigms described above leads to the following relationships:

$$MP^{traditional}_{product\_quality\_assurance.} \prec MP^{e-Measurement}_{e-Control} \prec MP^{AOSE}_{agents}$$

based on the internal (in-house) measurement and improvement and considering the training phase in MAS development as

$$MP^{traditional}_{project\_controlling} \prec MP^{e-Measurement}_{e-Quality} \prec MP^{AOSE}_{MAS\_development}$$

which can be characterized as a *moderate measurement process improvement*.

3.7. Telemetry based Software Measurement

One approach to overcome disadvantages of monolithic tools was the invention of telemetry based software measurement.

As an instrument for monitoring and control telemetry has been used over time in a lot of different fields of application (e.g. oil pipeline operation, production processes, and aerospace industry). The Encyclopedia Britannica defines the terms as: *„a highly automated communications process by which measurements are made and other data collect data remote or inaccessible points and transmitted to receiving equipment for monitoring, display, and recording.” [EncBrit08]*

Mapping the idea to software measurement means that an automation of measurement data capture, collection, and measurement data analysis should be achieved. Furthermore, the measurement process should not interfere the activity of a developer.

In the area of software measurement telemetry-based measurement has to fulfill four characteristics [JohKou+05]:

- Measurement data has to be captured automatically and non obstructive
- Measurements represent events which contains a timestamp
- All stakeholders have continuous access to the data
- Analytics enable short time controlling and forecast

The implementation of the telemetry approach requires the application of measurement sensors. For the application in means of telemetry based software measurement a measurement sensor can be defined as: a software program which is integrated into a software development tool (CASE-tool), whereas multiple sensors can be integrated into one tool. The sensor thereby is intended to capture measures, collect measurement data and to transmit measurement data to analyzing tools or measurement data storage facilities.

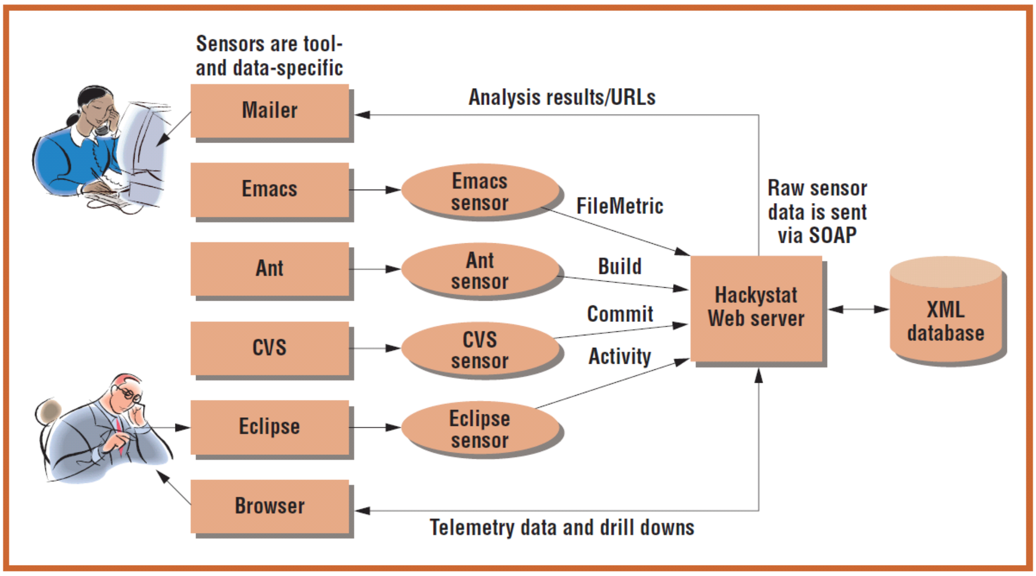
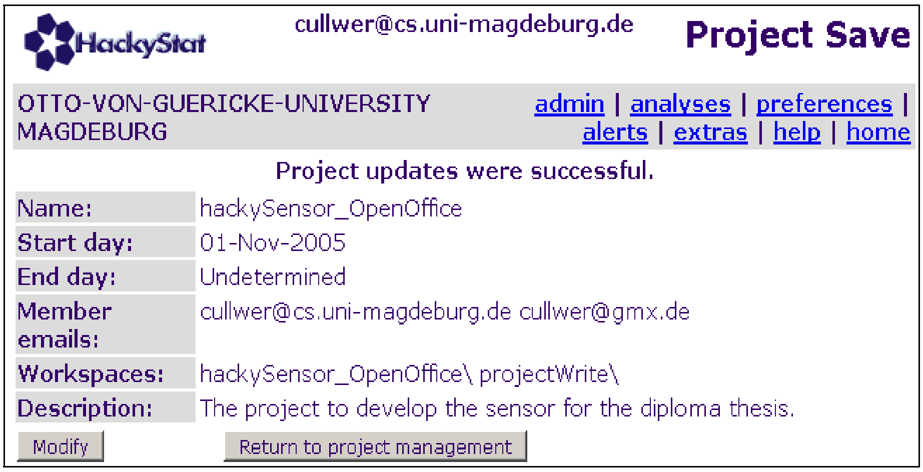


Figure 49: basic architecture from the Hackystat framework [JohKou+05]

Especially the availability of the measurement data for all stakeholders requires a different approach for the desired infrastructure because often used measurement standards defines three different views on software measurement (technical, strategic, and tactical view) (see chapter 2.3.3.). Furthermore the telemetry approach bears differences regarding software management and constitutes three management requirements [Johnson01]:

- *Not disruptive*: no detraction, overhead or context swapping due to data collection
- *Developer oriented*: measurement and data collection direct out of the developer activities and analysis are oriented to this activities
- *In-process*: analysis and evaluations are embedded into the development process.



cullwer@cs.uni-magdeburg.de

Project Save

OTTO-VON-GUERICKE-UNIVERSITY MAGDEBURG

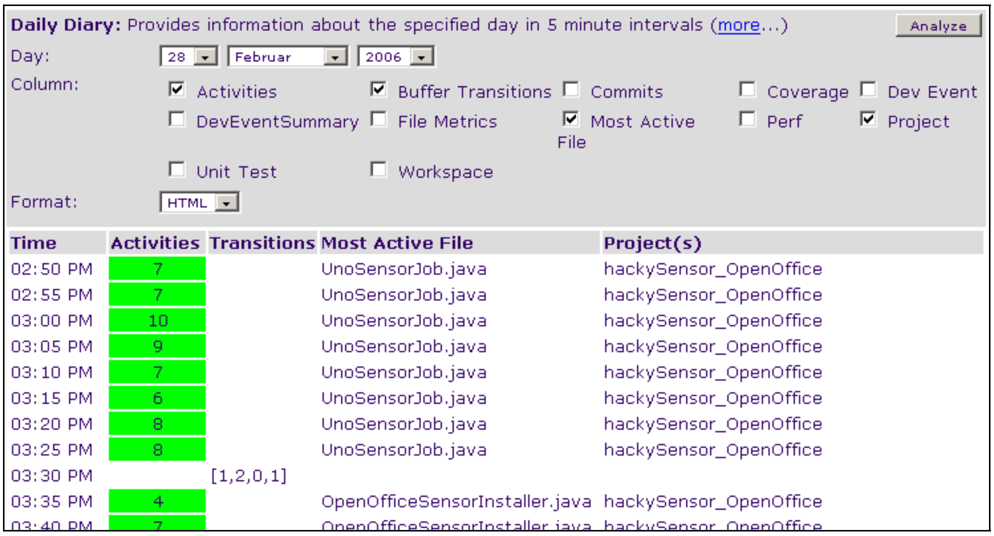
admin | analyses | preferences | alerts | extras | help | home

Project updates were successful.

Name:	hackySensor_OpenOffice
Start day:	01-Nov-2005
End day:	Undetermined
Member emails:	cullwer@cs.uni-magdeburg.de cullwer@gmx.de
Workspaces:	hackySensor_OpenOffice\ projectWrite\
Description:	The project to develop the sensor for the diploma thesis.

Figure 50: Hackystat sensor definition [Ullwer06]

A practical use case for example can be the measurement of changes in design documents. Editing, printing, and expanding of interface definition in specification documents can be measured for instance by a Hackystat sensor. A project at the Software Measurement Lab at the University of Magdeburg has been established to prove the capabilities of the Hackystat framework in this regard.



Daily Diary: Provides information about the specified day in 5 minute intervals (more...)

Day: 28 Februar 2006

Column: ☑ Activities ☑ Buffer Transitions □ Commits □ Coverage □ Dev Event □ DevEventSummary □ File Metrics ☑ Most Active File □ Perf ☑ Project □ Unit Test □ Workspace

Format: HTML

Time	Activities	Transitions	Most Active File	Project(s)
02:50 PM	7		UnoSensorJob.java	hackySensor_OpenOffice
02:55 PM	7		UnoSensorJob.java	hackySensor_OpenOffice
03:00 PM	10		UnoSensorJob.java	hackySensor_OpenOffice
03:05 PM	9		UnoSensorJob.java	hackySensor_OpenOffice
03:10 PM	7		UnoSensorJob.java	hackySensor_OpenOffice
03:15 PM	6		UnoSensorJob.java	hackySensor_OpenOffice
03:20 PM	8		UnoSensorJob.java	hackySensor_OpenOffice
03:25 PM	8		UnoSensorJob.java	hackySensor_OpenOffice
03:30 PM		[1,2,0,1]		
03:35 PM	4		OpenOfficeSensorInstaller.java	hackySensor_OpenOffice
03:40 PM	7		OpenOfficeSensorInstaller.java	hackySensor_OpenOffice

Figure 51: Activities over time analysis

The orientation on the developers view makes the telemetry-approach promising for specific product related measures and measurements. Unfortunately the framework is limited to Hackystat sensors and existing tools can only be included by data integration the functionality cannot be used separately and the framework contains no methods for the automated selection of appropriate sensors for a distinct information need (for example be the use of semantic descriptions),

For the analysis of measurement results the Hackystat framework uses a browser-based approach. The measurement data is stored on a public web server. That can counter the efforts for higher measurement acceptance due to security concerns because measurement results are normally a high valuable good in software engineering industry.

General shortcomings of Hackystat:

- Measurement is limited to Hackystat sensors, other approaches needs second parallel infrastructure
- Development of measurement sensors is driven by committed non-professionals and does not take into account professional information needs
- The framework does not take into account measurement standards
- Measurement is considered as enclosed task, needed integration into process frameworks is difficult

Summary: Telemetry-based Software Measurement

Unfortunately, in practice common software measurement tools find small acceptance due to their high costs, inflexible structures, missing integration capabilities, and therewith unclear cost/benefit ratio. That disadvantages have been tackled by e-Measurement approaches and by the definition of telemetry-based measurement.

Subsuming a combination of all approaches seems to be a promising approach for the creation of a measurement architecture providing tailored functionality by using measurement services, containing semantic described measurement data and experience, and building an infrastructure by applying e-Measurement characteristics and using ICT.

3.8. Measurement Paradigms Evaluation

This chapter discussed the software measurement involvements and different levels addressing different software technology paradigms such as Web-based software engineering (WBSE), agent-based software engineering (AOSE) and service-oriented software engineering (SOSE) [DumkeKunz^{+}08]. Based on these technologies an infrastructure-based measurement service was discussed considering the quality assurance themselves.

For an evaluation of the aforementioned paradigms a lot of different aspects have to be taken into account. To provide a visualization of all aspects in one figure an adoption of the kiviat diagram type using an interlacing approach has been chosen.

The following figure summarizes the different aspects of measurement process evaluation considering the best at the outer circle.

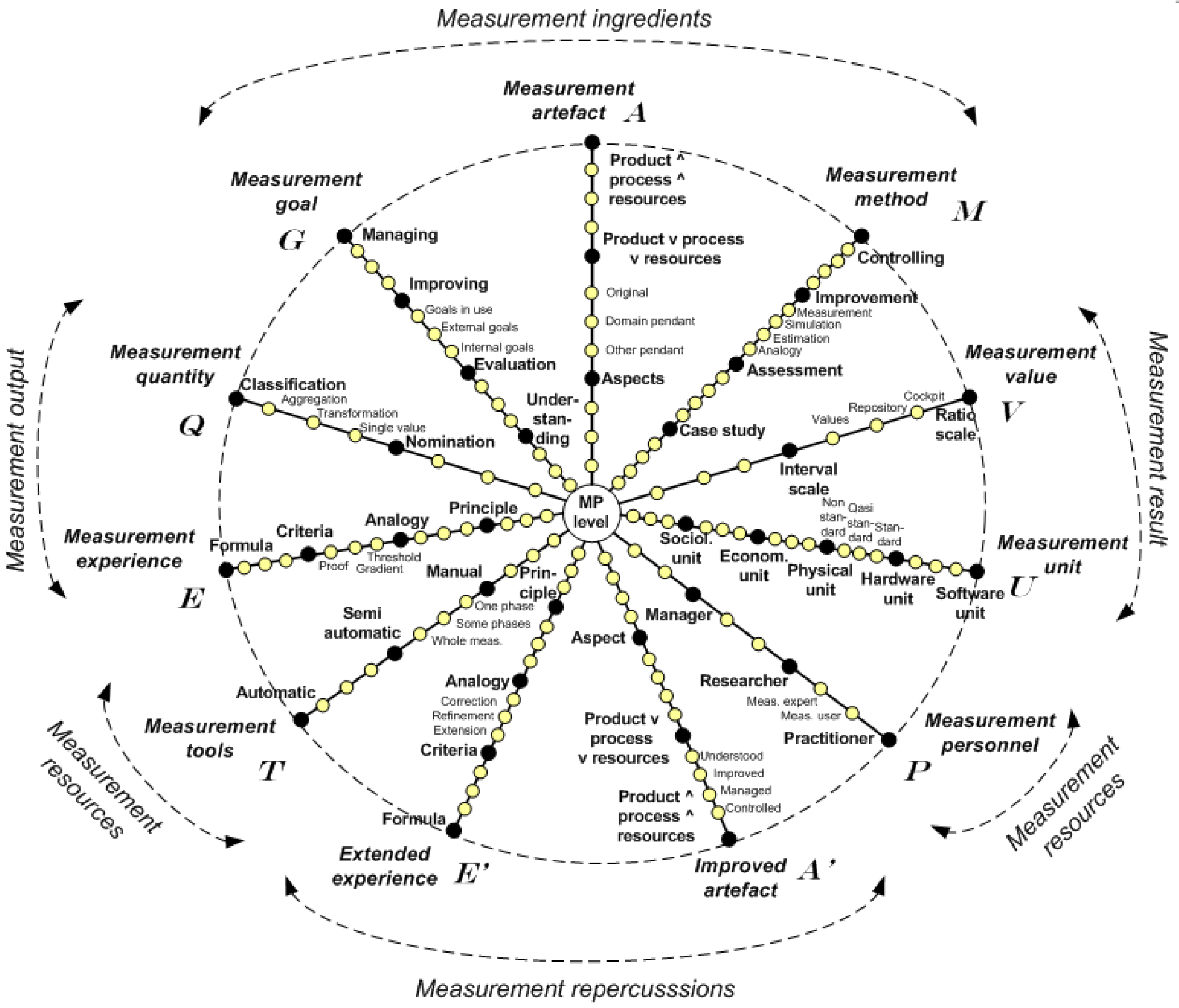


Figure 52: Software measurement process aspects and levels

Note the shown sub characteristics in this chart are described only one time per measurement component.

Based on this kind of visualization we can demonstrate the different levels of measurement processes in the following manner. Figure 53 compares the measurement process levels of an example of traditional measurement, e-Measurement and AOSE.

Taking into account the aforementioned formal considerations, the resulting levels representing ordinal scale type. The main reason is that the previous analysis only contains ranking order.

As shown in Figure even the analysis of measurement paradigms based on ordinal scale type provides valuable results which drawbacks of existing solutions have to mentioned and which levels a measurement infrastructure should cover to provide a useful approach.

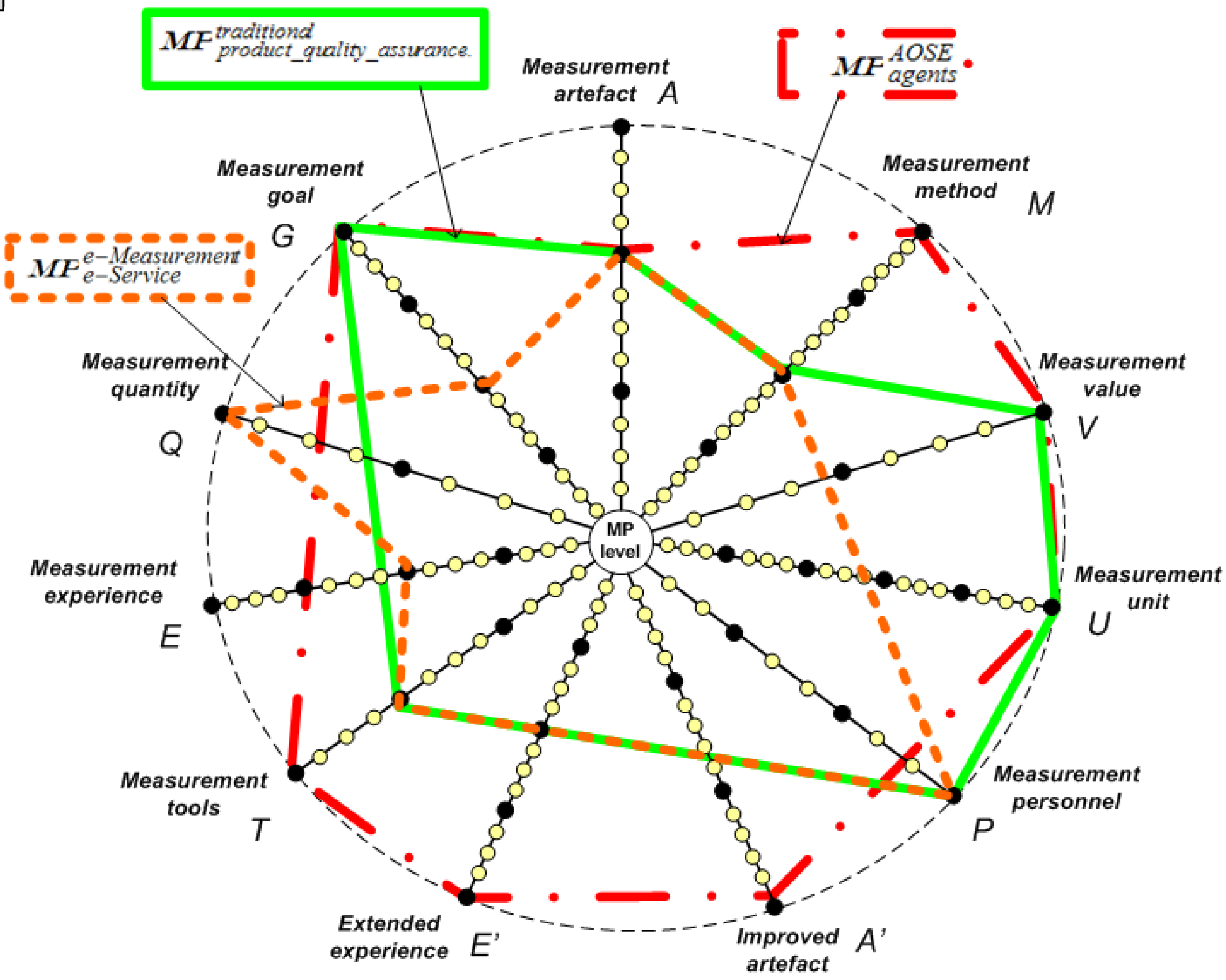


Figure 53: Software measurement process levels

Taking a closer look to the results in Figure 53 it is obvious that there is no single approach which combines outstanding characteristics in every relevant category.

In comparison to traditional software measurement, e-Measurement and Agent-oriented measurement delivers good results in most categories. Particular Agent-oriented measurement is outstanding in some categories

The consequential limitation constitutes the need for building infrastructures from scratch determine effort to avoid technological isolated solutions especially regarding the capabilities to integrate existing functionality and data [DumkeKunz$^+$06a].

Their major drawback, the missing practical usage, leads to other approaches for integration technologies. One of the most observed in recent surveys is the approach of service-oriented architectures [Schmietendorf07]. The idea of the approach and their capabilities for building software measurement infrastructures is presented in the next chapter.

4. SOA-based IT Architectures

4.1. Introduction

After describing the state of the art and pointing out techniques and technologies that could depict the way to next generation measurement tools, the general question arises if one should start from the scratch with something complete new or is upgrading and enhancing of existing solutions the better choice.

Observations in related IT areas show that the integration of existing solutions in new approaches is often necessary to promote the acceptance of new implementations by assuring former investments and enable a stepwise migration.

Mapping this idea to software measurement tools changes the viewpoint from product-centered to process-centered software architectures [FarBraKunz+06].

A general definition of software architecture can be found in [BassClements+03]: “The software architecture of a program or computing systems is the structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.”

Service-oriented architectures (SOA) as the prevailing approach and upcoming industry standard for integration and creation of corporate IT architectures has been chosen as the technological groundwork for the thesis in hand [RuSchmKunz+07a].

The following chapter is intended to describe the concept and uses technologies of service-oriented architectures in a nutshell.

4.2. Aspects of Service-oriented Architectures

One of the biggest challenges in the software-engineering field is the paradigm change from “from scratch” development without taking into account existing implementations to reuse striving software development processes were the focus lies on the development of functional services to establish service oriented architectures [Meinel06].

In difference to this definition of software architectures service-oriented architectures are not restricted to a single program or system. Contrariwise they deal with company-wide or as the case may be with cross-company IT system architectures.

A service-oriented architecture has to have an integrated process-oriented perspective of IT system architecture. In the foreground of a service-oriented solution is not a single application but the integration of different components which provide their functionality as services.

Therefore a multi-tier integration architecture for a service-oriented architecture is needed. Shan and Hua proposed a solution named SANTA (Solution Architecture for N-Tier Applications) [ShanHua06]. This model was mapped to SOA in [Schmietendorf07]. The main contribution is the determination of different layers for business process, basic services and existing applications.

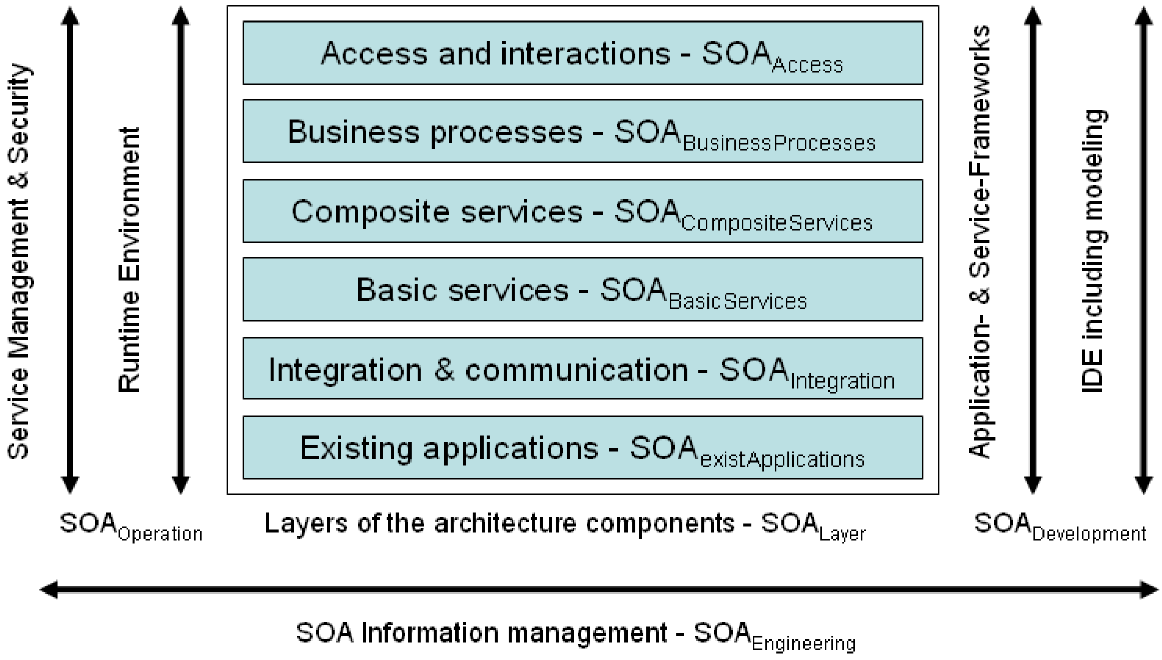

Access and interactions - SOA_{Access}
Business processes - $SOA_{BusinessProcesses}$
Composite services - $SOA_{CompositeServices}$
Basic services - $SOA_{BasicServices}$
Integration & communication - $SOA_{Integration}$
Existing applications - $SOA_{existApplications}$
Service Management & Security
Runtime Environment
$SOA_{Operation}$
Layers of the architecture components - SOA_{Layer}
Application- & Service-Frameworks
IDE including modeling
$SOA_{Development}$
SOA Information management - $SOA_{Engineering}$

Figure 54: Service-oriented integration architecture [Schmietendorf07]

Existing applications provide their functionality by using basic services or composite services respectively to execute assigned business processes [RuSchmKunz+07b].

Such services can be spread across heterogeneous IT landscapes by being implemented in different programming languages or they can only be integrated temporarily. But only as recently as a standard technology for the implementation of services arises SOA solutions found more and more acceptance. Especially the creation of the eXtensible Markup Language (XML) and the usage of this language to describe services, relationship of services and service interfaces forced the application of service-oriented architectures from the technological side.

This technological viewpoint marked the consideration of service-oriented architectures in the beginning of its appearance. This viewpoint was established in the definition of SOA by the Gartner Group in 2003:

"Essentially, SOA is a software architecture that builds a topology of interfaces, interface implementations and interface calls. SOA is a relationship of services and service consumers, both software modules large enough to represent a complete business function. Services are software modules that are accessed by name via an interface, typically in a request-reply mode. Service consumers are software that embeds a service interface proxy (the client representation of the interface)."[Natis03]

With raising application and propagation and the determination in different layers (see Figure 54) the definition what a SOA is about shifts from the technological perspective to a methodical one where SOA is defined as a paradigm:

"Service-oriented Architecture is a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains." [MacKenLas$^+$06]

4.2.1 Demarcation against other Integration Proposals

The idea of service-oriented architectures has not been emerged with the development of the web service technology [DostalJeckle04]. The basic concepts were presented in the surrounding of distributed communication models like DCE (Distributed Computing Environment) or CORBA (Common Object Request Broker). Both approaches were focused on client/server-based systems and to a less extent on the implementation of company-wide integration architectures.

But in difference two this two techniques there are approaches for company-wide integration solutions in the context of Enterprise Resource Planning systems (ERP) and in the Enterprise Application Integration (EAI) approach [Schmietendorf07].

Integrated ERP systems offered for example by SAP or Oracle provide a consistent and homogeneous data model to realize integrated and unified business processes on an overlying level. They apply data and process integration to provide a consistent presentation [Woods04].

Due to the concentration of ERP systems the approach of enterprise service architectures is not an alternative the area of software measurement infrastructures.

4.2.2. Technological Aspects of Web Services

Service offers provided by service-oriented architectures are mostly characterized by attributes that appear in the context of component-based and object-oriented software engineering. Additionally some new characteristics emerged such as the unavailability of the source code in most cases and the fact that a service can be already running at the point in time when the service is integrated in new developed architectures.

In general the characteristics of services can be sorted into two different aspects [Erl05]:

Technical characteristics

- A service provides functionality across well defined interfaces. The internal details of the implementation are hidden.
- Services support a loose coupling. That means that modifications in one service not entail modifications in other services
- Services are executed autonomic. The needed and used resources are controlled by the service itself.

- Services support reuse in principle. Therefore design guidelines have to be established as mandatory.
- Services are stateless. Information is stored only for a specific session. Mechanisms for the storage of a status are allocated at a higher level.

Functional characteristics

- Services can be assembled by using existing applications or services. New requirements can be implemented by the orchestration or choreography of existing services.
- The usage of a service implies an agreement about functional and non functional requirements.
- Services can be discovered on demand by human or technical users.

Even if it is only one possible way of implementation, service-oriented architectures are linked most times with web services. That is the case because the technology of web services progresses the implementation of SOA's very much.

According to the definition of the World Wide Web Consortium (W3C) a web service is a *"software system, designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-process able format."* [BoothHugo^{+}04]

The cornerstone for the pervasion of web service technology was placed by the prevalence of XML with its advantages of platform independence and readability for machine and human users.

XML by itself is not only a mark-up language in contrast to the name. In fact it is possible to define new languages by using the contained rules and methods. Especially the separation of the content and the logical structure of a document to describe the enclosed data is a key target. In this way the term "meta-data" was inducted to depict the fact that enclosed content is described with a logical structure.

A language defined with XML consist a number of mark-up tags and a document specific tree structure. The combination of both is called content model and stored as Document Type Definition (DTD) or XML Schema (XSD).

The usage of XML can provide some advantages in the area of service-oriented architectures. XML can be used as a universal exchange format data description and interface definition to enable communication between applications or services. XML offers the possibility of using enclosed meta-data to describe the document structure with its hierarchy and data elements. But the XML file and the corresponding scheme definition is linked by loose coupling which enabled platform independent implementation [ChaBer08].

Thus, four key attributes of web services can be declared [KosLey04]:

- a web service can be identified by a Uniform Resource Identifier (URI)
- the web service interface is machine readable and described in a standardized way
- a web service communicates with other software systems through XML messages
- a web service acts autonomously, a distinct observation in which way a message is processed is not intended

This key attributes direct to a so-called web service protocol stack with different technologies and specifications for the different attributes. The several building blocks are described next.

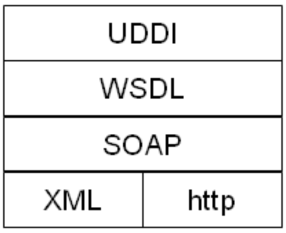
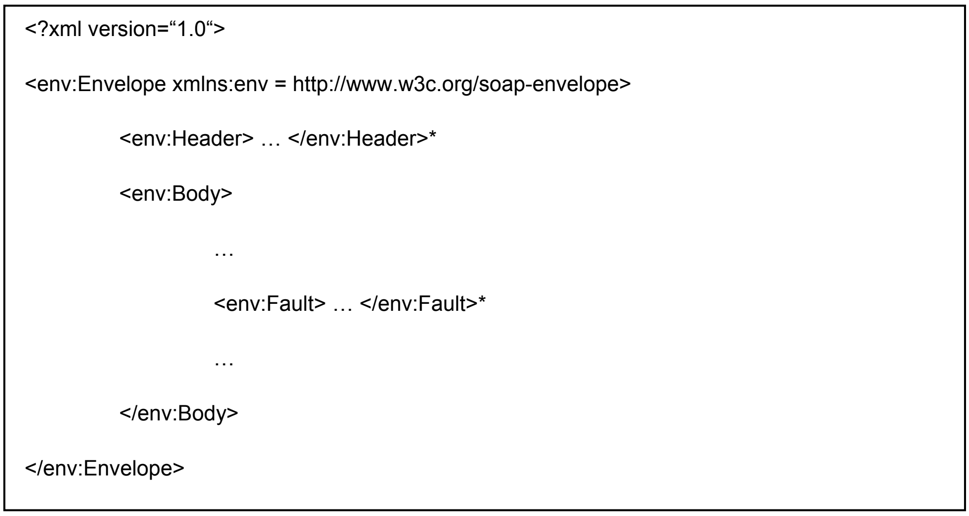


Figure 55: Basic web service protocol stack [KosLey04]

The communication between web services (which consists method invocation, parameter and result transfer) is realized by the Simple Object Access Protocol (SOAP). Even if the usage of the Hypertext Transfer Protocol (HTTP) is not mandatory SOAP is often summed up by the formula *SOAP = XML over HTTP* [KnuDim+03].

A SOAP message consist of a so called BODY with data and an optional HEADER with additional information for example how the data is coded or which authentication is used.



```
<?xml version="1.0">

<env:Envelope xmlns:env = http://www.w3c.org/soap-envelope>

        <env:Header> … </env:Header>*

        <env:Body>

                …

                <env:Fault> … </env:Fault>*

                …

        </env:Body>

</env:Envelope>
```

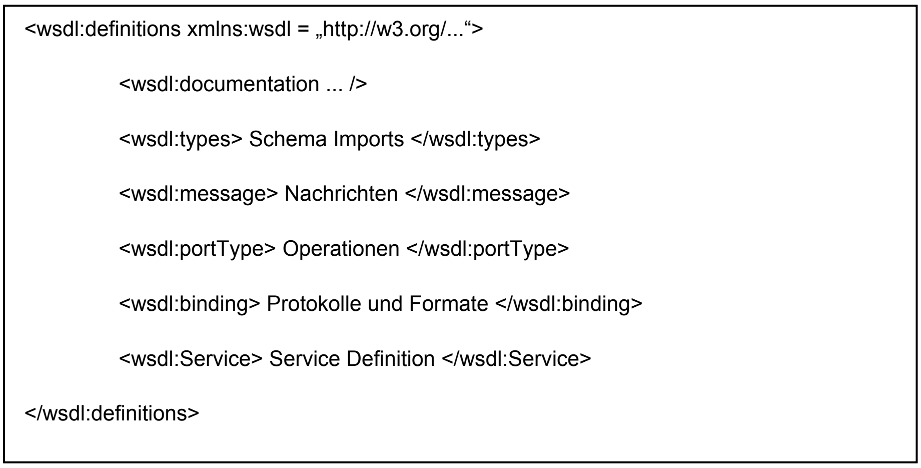
Figure 56: Structure of a SOAP message

According to [Short02] the usage of SOAP provides some advantages in the area of service-oriented architectures. As explained above SOAP is independent from specific transport protocols. HTTP is specific but since a SOAP-message is just a XML file it can be transferred with other protocols.

Existing middleware systems like J2EE (Enterprise Java Beans) or COM+ (Component Object Model) can be modified to support SOAP. In this way SOAP enables interoperability between different middleware approaches.

And as most important advantage SOAP is programming language and platform independent. Every technology that supports XML and HTTP can interact with other technologies doing the same.

Having described the structure of messages the next important point is the definition of interfaces. The Web Service Description Language (WSDL) is used for the XML-based interface definition for web services.



```
<wsdl:definitions xmlns:wsdl = „http://w3.org/...“>
        <wsdl:documentation ... />
        <wsdl:types> Schema Imports </wsdl:types>
        <wsdl:message> Nachrichten </wsdl:message>
        <wsdl:portType> Operationen </wsdl:portType>
        <wsdl:binding> Protokolle und Formate </wsdl:binding>
        <wsdl:Service> Service Definition </wsdl:Service>
</wsdl:definitions>
```

Figure 57: Structure of a WSDL description

This WSDL definition accordingly describes mainly technical aspects to enable the implementation and not quality or commercial aspects.

The WSDL definition addresses three major questions:

1. What offers the service? That defines what messages that are sent by the service and which operations are offered for clients of this service.

2. How are the messages defined? That defines which protocols are used by the web service to encode the messages (afore cited SOAP is the standard protocol but other protocols can be used to).

3. Where the web service located? Typically the name and physical address (e.g. Unified Resource Identifier) is defined here.

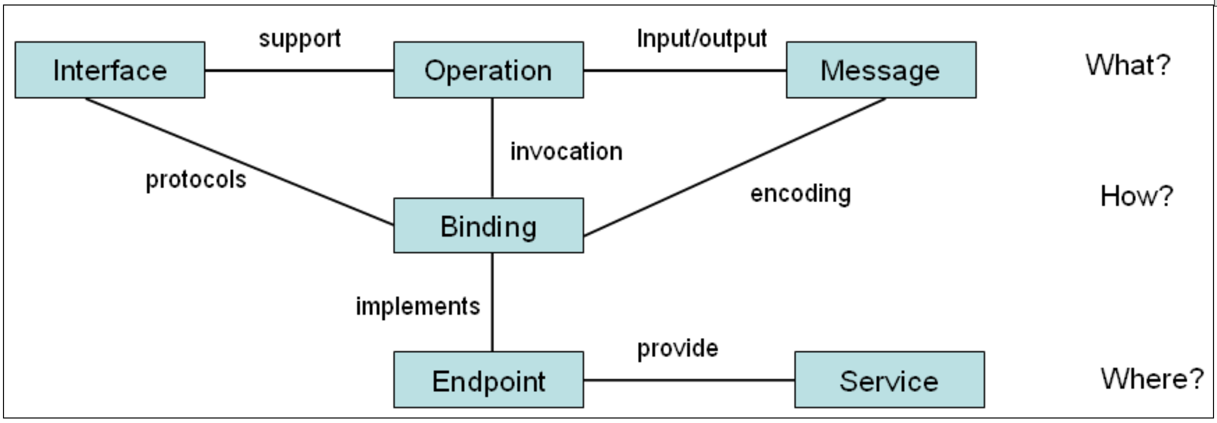


Figure 58: Overview about WSDL concepts [KosLey04]

Figure 58**:** Overview about WSDL concepts [KosLey04] presents an overview about the WSDL concept and the three described aspects.

Another basic technological specification for services is UDDI (Universal Description, Discovery, and Integration). This specification is used to identify and discover widespread web services. The main goals are cataloguing, indexing and registration of web services. Therefore the contained information is distinguished into White-, Yellow-, and Green-Pages.

White- and Yellow-Pages contain information about the service provider (contact information and industry sector) and the Green-Pages contain the technical description of the service (basically the WSDL definition). The fundamental application of UDDI is shown in Figure 59

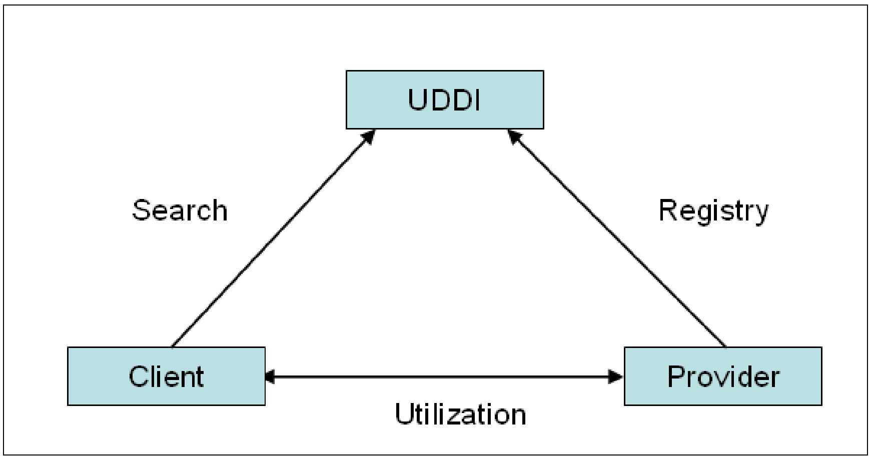


Figure 59: Basic application of the UDDI concept [KosLey04]

From technical viewpoint UDDI provides an expandable data model for the description of web services and possible multiple client-side or server-side interfaces which support the registry, the search and the administration of services.

In contrast to other integration approaches SOA enables to assemble different service activities. In this context the terms orchestration and choreography have been stamped to describe the different way of cooperation of web services [DostalJeckle+05]. Thereby orchestration describes the executable aspects of a business process from a process viewpoint with a central controlling instance. On the other hand choreography describes the tasks and the interplay of different processes with a peer-to-peer characteristic.

The goal of orchestration is to map a business process to a sequence of atomic service implementations. The responsibility for the process execution and the execution of the behind services is assigned to a single stakeholder (who is in charge for the single business process) [Erl05]. Whereas, the choreography approach combines the process executions of different stakeholders and realizes an intercorporate-wide process handling [Erl05].

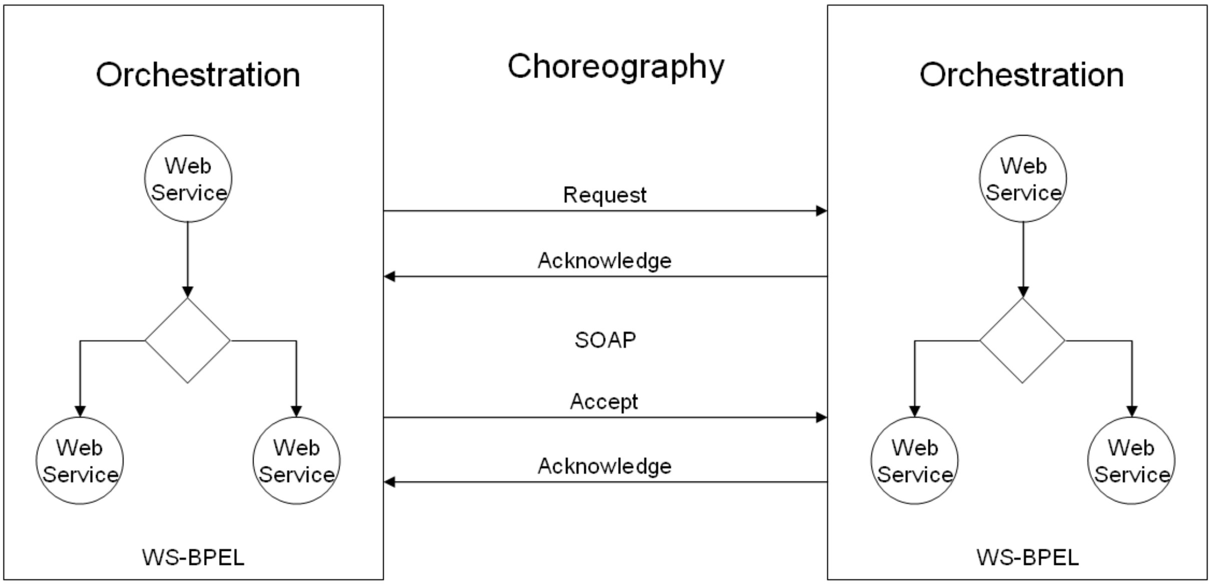


Figure 60: Principle of orchestration and choreography [Peltz03]

For the two approaches different web service technologies has been established. For orchestration the Business Process Execution Language for Web Services (BPEL4WS) and for choreography the Web Service Choreography Description Language (WS-CDL). Due to the fact that BPEL has become a de-facto standard and under consideration of the drawbacks of existing choreography solutions [BarrosDumas$^+$05] the explanations are constrict to orchestration in the following.

BPEL4WS is a XML based programming language and enables the mapping, controlling and execution of business process by using of web services. Every business process can be defined as an abstract process (definition of process characteristics) or an executable process (definition of internal implementation).

With BPEL it is possible to define which information are transferred to a specific destination and which functionality is called in a specific location. Additionally exceptions (for example error processing) can be defined. BPEL enables the user to define specific sequences of web services without taking care of their internal implementation.

Thus, for every process or sub-process the following attributes are defined [AlvesAssaf+07]:

- Partner links. Connection of stakeholders to WSDL-port types
- Variable declaration by using XML schema
- Definition of events, compensation, fault, and termination handlers
- Description of the desired process action

For the description of the desired process action a set of atomic (*e.g. invoke, receive, reply, assign, throw, validate, exit*) and structured activities *(e.g. process, scope, sequence, repeatUntil, forEach, if, flow*) are predefined.

For the definition of BPEL documents often GUI based tools (e.g. Active BPEL Designer) are used but they can also be generated from graphical notation (e.g. Business Process Modeling Notation). The execution of defined processes is done by so-called WS-BPEL engines [RudKunz+07].

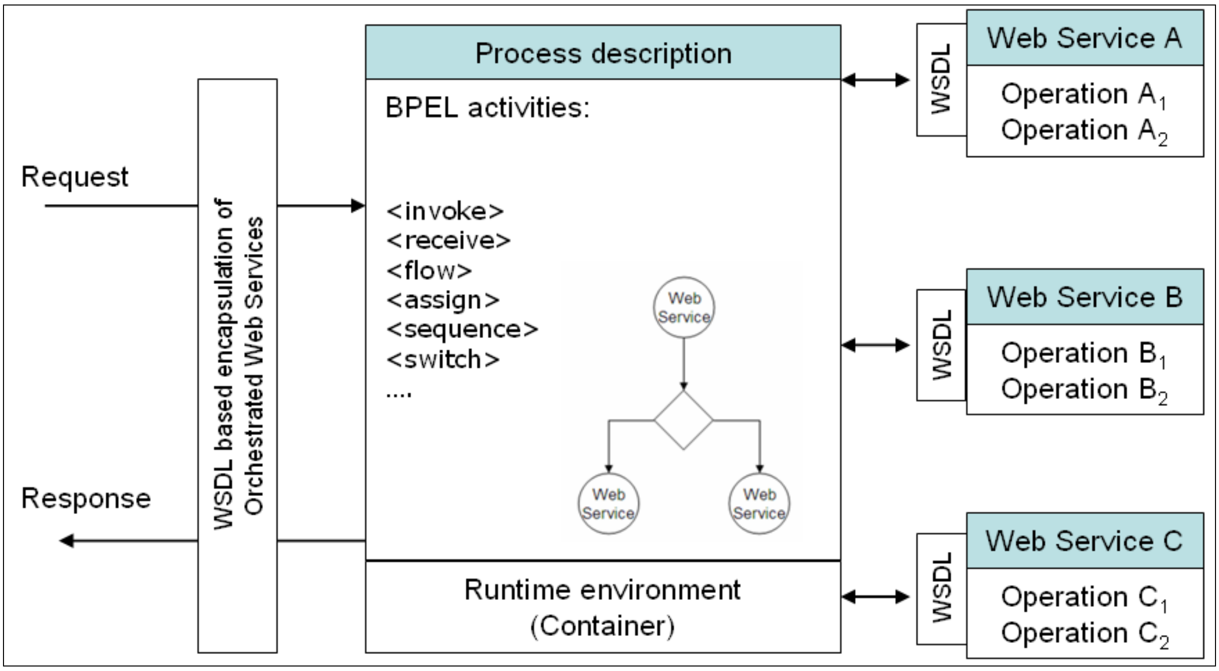


Figure 61: Architecture of a BPEL environment [Collaxa03]

Summary: Service-oriented Approach

EAI solutions are able to realize business processes by using applications or data source over a network. Thereby they are able to implement company wide architectures. Such solutions often only provide a technical view on integration while describe in the core a platform for message exchange, protocol conversion, and mapping of information via a bus system [ReiSchm+03].

By contrast a Service-oriented architecture provides integration on a functional dimension and thereby a decoupling of applications. In that case not applications are used but functionality is invoked. By using standardized service descriptions, services can be exchanged without affect the execution of superordinated business processes.

Only for the integration of existing measurement databases EAI techniques should be taken into account to provide as much empirical knowledge as possible for any Service-oriented architecture.

Currently the Web-Service technology is used mainly within the application development as substitution of proven middleware technology as for example CORBA. The arising advantage consists of the resulting technology independence. In this way a combination of techniques like J2EE from SUN or .net from Microsoft is possible [Schmietendorf07]. Through the standardization Web Services can be used in manifold use cases they can be integrated just in time for various business processes far beyond the boundaries of the individual enterprise.

4.3. SOA-capability of Software Measurement Tools

The analysis of existing measurement tools was performed with two different viewpoints:

a) A survey of manufacturers about existing products and existing integration opportunities

b) An independent analysis of existing measurement tools concerning the SOA-capability

The presented detailed analysis covers aspects concerning among others functionality, interfaces, security, payment options, and supported standards [SchmKunz+07].

As an additional result three major capability levels of service interfaces has been identified:

1) Output of measurement values
2) Input of measurement data
3) Control and triggering of measurement services

In this way we consider two different views for interface evaluation: the used technology and the provided functionality.

4.3.1. Assessment about SOA-capability of measurement tools

The target of this section is to analyze how far the aspects of service-oriented architectures have penetrated into the area of software measurement tools.

The SOA-capability depends predominantly on import potentiality of measurement data, export potentiality of results, and the implemented facilities to control the tool.

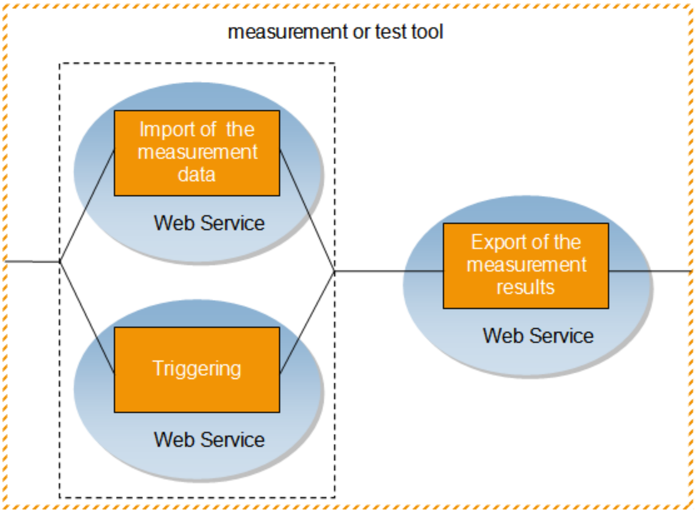


Figure 62: substantial functionality for SOA-capability

Since the most important thing for the usage of a distinct tool is the occurrence of interfaces for the export of measurement results, the analysis is focused on this aspect.

At first, a classification of interfaces according to the needed effort to adapt the interfaces into a service-oriented measurement infrastructure is introduced. And five major categories are identified and pointed out in the following table.

Table 15: Effort classification for SOA-execution of measurement results

Existing Web-Service-Interface	*Very marginal effort*
XML-based Interfaces	*Low effort* (simple creation of web-services out of XML-files and databases)
Standardized exchange format	*Effort at medium level*
Proprietary exchange format	*Medium up to high effort*
No exchange possibility (only paper and printout)	*Continuous effort due to manual information transfer*

By applying this classification approach to the 39 analyzed tools, one can see that over half of all tools just have a proprietary interface with a medium up to high adoption effort. If a measurement tool provides more than one type of interface the one with the lower effort was captured. The results are pointed out in Figure 63 while the detail results for each tool are put into the appendix.

Based on the results one can reason that the majority of existing measurement tools have a medium or high effort for embedding them into an SOA. Only nearly one third (31%) need just marginal or low effort.

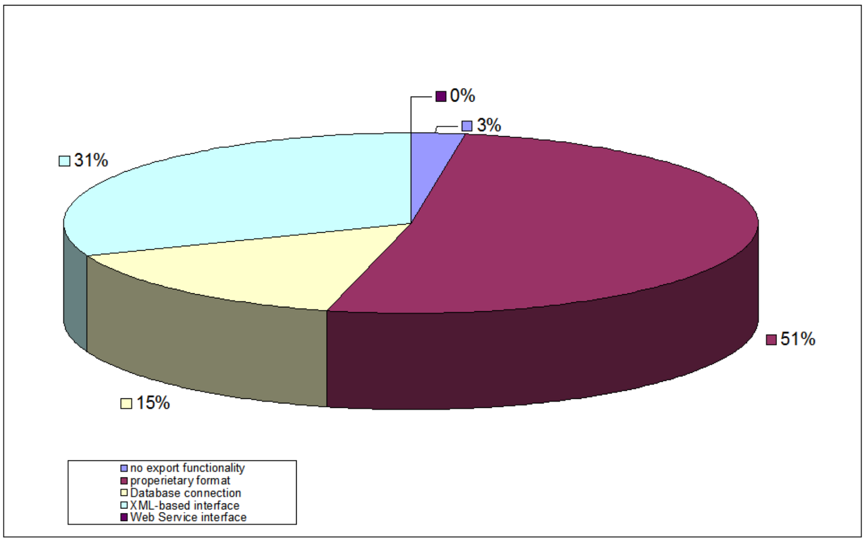


Figure 63: Occurrence of different interface technologies

The second focus in the survey was on the combination of the functionality of different tools and/or the possibility to control the tool by an interface. This is important for the SOA idea of measurement tools because the main goal is to create a tailored measurement tool out of the functionality from different independent tools.

The results in this area are very uneven. Over half of all the tools just provide proprietary interfaces and in this way no possibilities exist to access functionality or to control the tool via the interface. Another 10% use database connections to transfer data in the first instance. These connections also do not have the precondition either to control or to access tool functionality.

Even if the other 34% do not provide options for controlling or accessing functionalities, the usage of XML at least allows enhancements in this direction in the near future.

As a subsumption one can say that the functionality of the presented interfaces does not deliver the needed requirements for service-oriented architectures. In this way the realization of a software measurement tool as a web service offer is the exception [NewLom03].

Therefore, the second step was to make a survey among measurement tool manufactures how far their products are applicable to create service-oriented measurement infrastructures, and if they are available as a service offer, respectively.

Another facet was to find out if there are any planed ideas for SOA´s in future releases, if at the moment there is no service-oriented functionality.

4.3.2. Survey among Measurement Tool Manufactures

As an outcome of our survey, over 30 replies has been received. Some of the tools were part of our first assessment and, in general, one can outline that the answers give a representative view on software measurement tools from the manufactures point of view.

The first look is again on export interfaces. The results are shown in Figure 64 and they are corresponding to the first assessment. Every tool has an export possibility but the effort to integrate them into SOA´s is not marginal due to non service-able connection type.

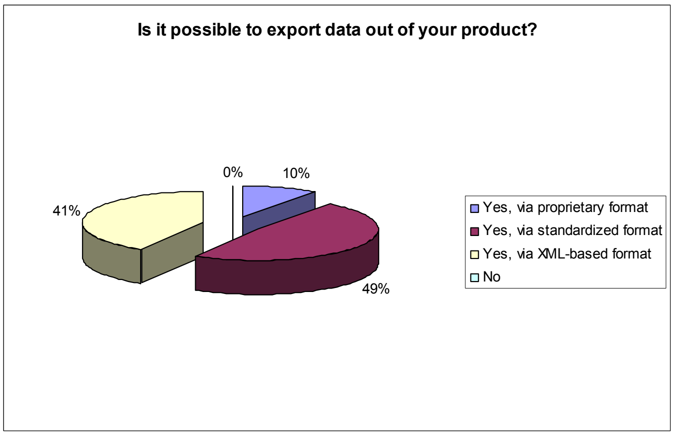


Figure 64: Results about export interfaces

The next questions target another important fact for Service-oriented Measurement Architectures: the combination of different tools according to distinct functionality. The results can be that for every distinct information need, different functionalities from different tools were combined to a new service by using web service orchestration [Peltz03].

The results point out an optimistic view to this issue: over 80% of all manufactures say that their tool can be combined with other tools (see Figure 65) and nearly 80% say that single functionality of their tools can be used independently (see Figure 66).

As aforementioned an approval of this appraisal is very doubtful. The first reason is that tool manufactures often understand combining opportunities as the option to combine tools from their product portfolio and not to other tools, for example by using open or standardized interfaces or connections. Because of the fact that more features have been implicated into the concept of tool combination, the independent assessment is not rebutted.

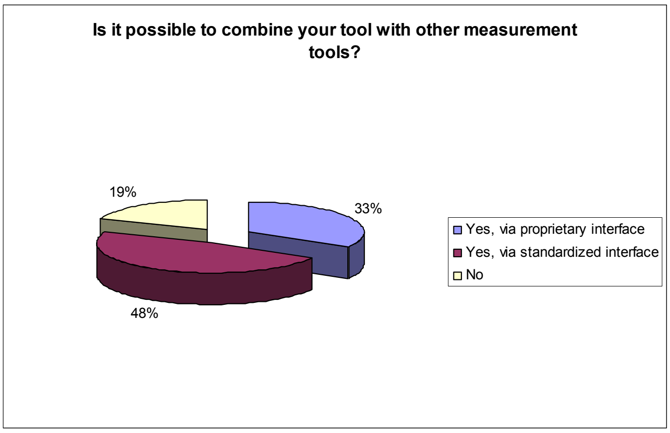


Figure 65: Combination of different measurement tools

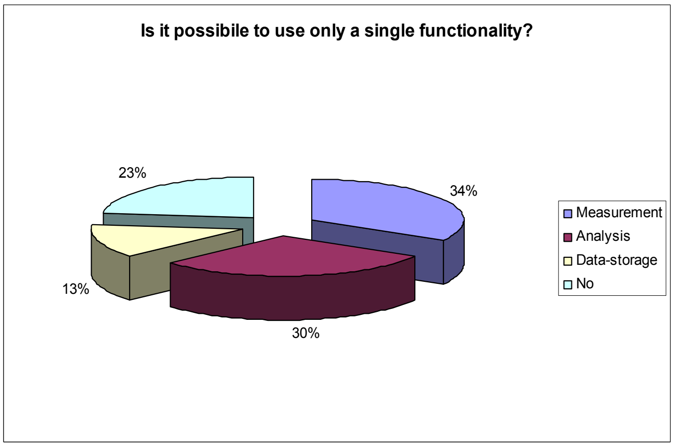


Figure 66: Usage of single functionality

Especially for commercial software, the question about the different type of license models is very important. If a software measurement tool provides functionality as a web service, the license model should provide pay-per-use to avoid high acquisition cost and therewith open up new possible users. Unfortunately this aspect is not considered by the tool manufactures at the moment [Kunz+06a].

Only 3% provide a pay-per-use option and just a quarter provide at least the possibility to purchase a single module/component.

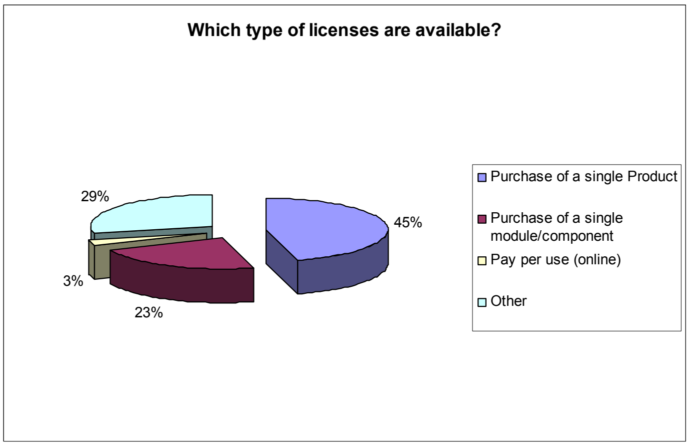


Figure 67: Different types of licenses models

Summary: Current Situation of SOA-based Measurement

Subsuming the assessment and the survey one can say that existing measurement tools are in general SOA-capable even if there is some effort to invest on.

But the establishment of new license models for Service oriented tools is necessary in view of the current situation. But to create a real benefit of Service oriented architectures, the possibility of combination among different measurement tools has to be improved. Therefore, an accepted SOA Guideline which describes requirement and interface requirements has to establish and to create an applicable approach for Service standardization and collaboration. Tailored functionality and new license models in the area of software measurement tools especially for small and medium-sized businesses can boost the application of software measurement in this area [Kunz+06b].

5. Service-oriented Measurement Infrastructures

The analysis of existing measurement tools and the target of implementing a standardized measurement process make it reasonable to take into account the technological solution of service-oriented architectures and the paradigm of providing services rather than monolithic tools. The application of the mentioned technologies could create a measurement infrastructure which should be defined as:

"A Measurement Infrastructure is an adaptable, automated, and coherent substructure for technical and organizational securing of a Corporate Measurement Program, through the holistic usage of CAME tools, a persistent data storage in a measurement database, and a measurement data exchange by the use of well defined interfaces."

To distinguish between the terms architecture and infrastructure for the thesis in hand an infrastructure is defined as an implementation of a distinct architecture.

Applying the technologies and the paradigm of service-oriented architectures to software measurement one will early come to a point that a lot of different ways exist to implement a software measurement program by means of SOA. Taking into account the described basic characteristics of SOA and the outlined needs for a software measurement infrastructure, it is obvious that the framework cannot end up in only one distinct measurement tool. Contrariwise, the thesis in hand aims to present a general framework for creating a service-oriented infrastructure for various software measurement use-cases.

The previous chapter points out that different aspects have to be taken into account to successfully implement a software measurement program. Summing up, the following goals are defined for a service-oriented measurement infrastructure:

- No limitation to product measurement. Support of resource and process measurement and establishment of
- Support of corporate measurement programs
- Controlling and enhancement of measurement process level and continuous realization of a standardized software measurement process
- Overcoming of general measurement tool shortcomings
- Establishment and support of software measurement repository
- Inclusion of a measurement experience base
- Application of ICT and guideline for the interplay of tools of different providers
- Standardized approach for integration of different services
- Enable different license models
- Enable automated procedures for measurement service orchestration according to a specific information need

In order to reach the defined goals different systems and components are needed for very different functionalities. To provide an overview Figure 68 shows the different elements and how they are interacting together.

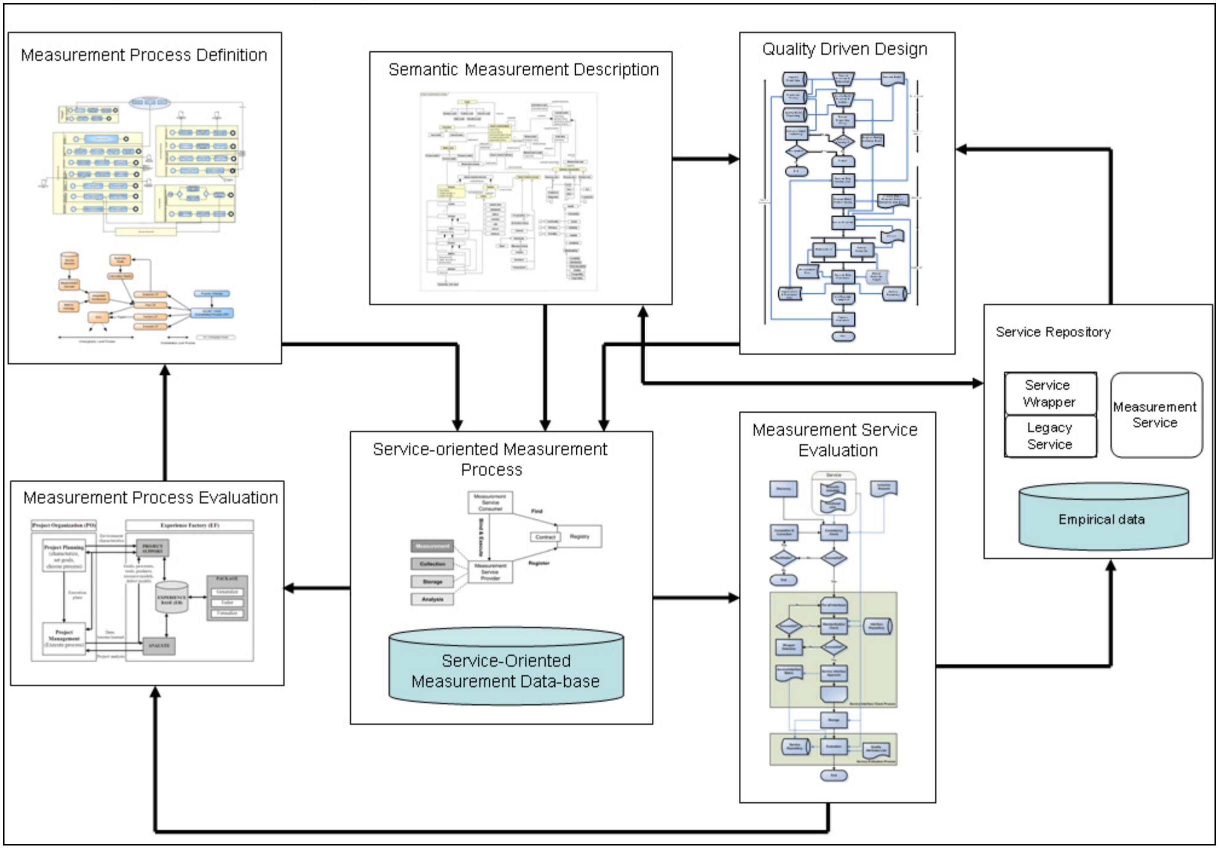

Measurement Process Definition
Semantic Measurement Description
Quality Driven Design
Service Repository
Service Wrapper
Legacy Service
Measurement Service
Empirical data
Service-oriented Measurement Process
Service-Oriented Measurement Data-base
Measurement Service Evaluation
Measurement Process Evaluation

Figure 68: Simplified architecture and included systems of the framework

Every decision for on a specific technology or on a specific subset of entities has been done with the target to create a coherent solution and not to cover all possible use cases.

The increasing economic relevance of software measurement for organizations cannot be neglected. But issues like complexity and missing traceability of measurement processes constitute the need for direction and guidance in this regard. This need is satisfied by the development of the Practical Software Measurement (PSM) [PSM01]. The main concepts of PSM provide a foundation for the international standard ISO/IEC 15939 (see Chapter2.3.3.3. The ISO/IEC15939 Software Measurement Process.).

As described in chapter 2 the ISO/IEC 15939 Information Technology – Software Measurement Process helps to identify, define, select, apply, and improve software measurement in any software development project. This standard describes a compliant measurement process concerning its purposes and outcomes combined with appropriate activities and tasks. Thereby the core measurement process is divided in the sub-processes “plan” and “perform” using the information needs as input, for example by using the Goal Question Metric approach (see chapter 2.3.3.1. The Goal-Question-Metric Method) with the aid of appropriate business goals.

For a better understanding the following figure draws the scope of the intended infrastructure is shown in context of the ISO standard.

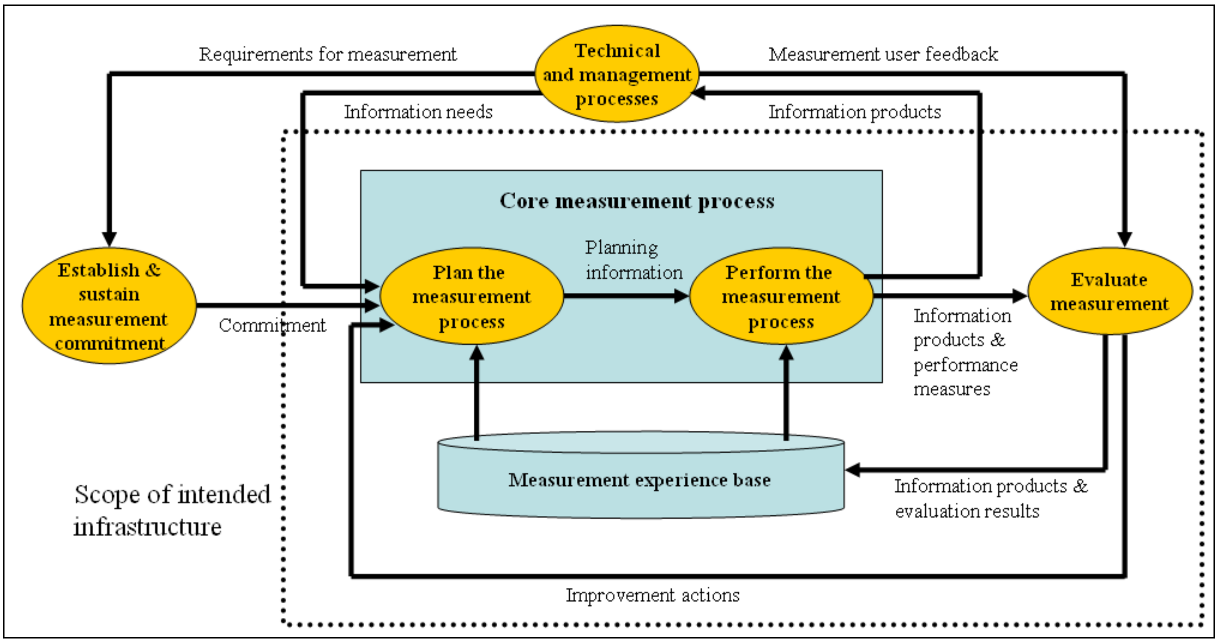

Requirements for measurement
Technical and management processes
Measurement user feedback
Information needs
Information products
Core measurement process
Establish & sustain measurement commitment
Commitment
Plan the measurement process
Planning information
Perform the measurement process
Evaluate measurement
Information products & performance measures
Measurement experience base
Information products & evaluation results
Scope of intended infrastructure
Improvement actions

Figure 69: Scope of ISO/IEC 15939 standard [ISO/IEC02]

To date, the core measurement process can be facilitated by monolithic software measurement tools taking a set of metrics and the measurement object as input and producing measurements together with some kind of evaluation or analysis [KernchenKunz$^+$07c].

Owing to high license costs of such monolithic and rigid measurement tools, the real benefit is hard to identify. Because of that the commercial application of software measurement seems to be increasable by using flexible infrastructures without high acquisition costs and with the possibility to tailor the functionality according to a project specific measurement information needs.

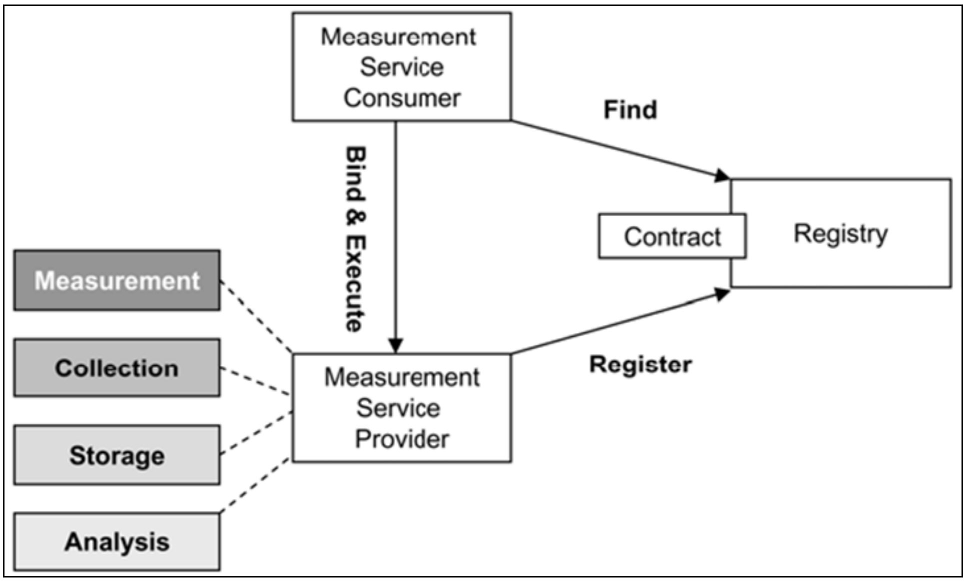


Figure 70: SOA-based Measurement Process

Based on the general characteristics of ISO/IEC 15939 a service-oriented measurement infrastructure with different services and components should be specified and implemented to realize the defined processes and activities.

To give an overview about the desired infrastructure Figure 71 describes the core architecture and the core elements aligned to the ISO/IEC 15939 measurement process standard

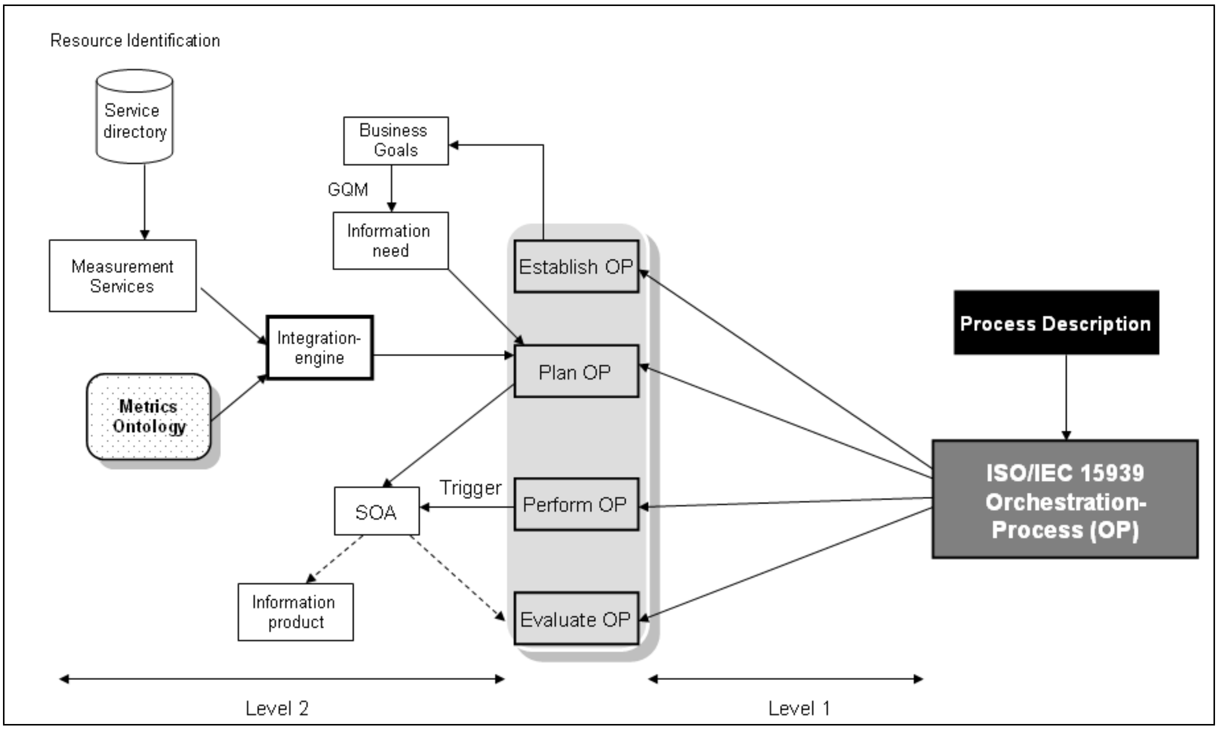


Figure 71: Different SOMI orchestration processes [Kunz+06f]

To create such an infrastructure it is necessary to define the technology or the notation in which the different elements or specifications have to be implemented or described. In this way a level-based procedure was used as shown in Figure 72.

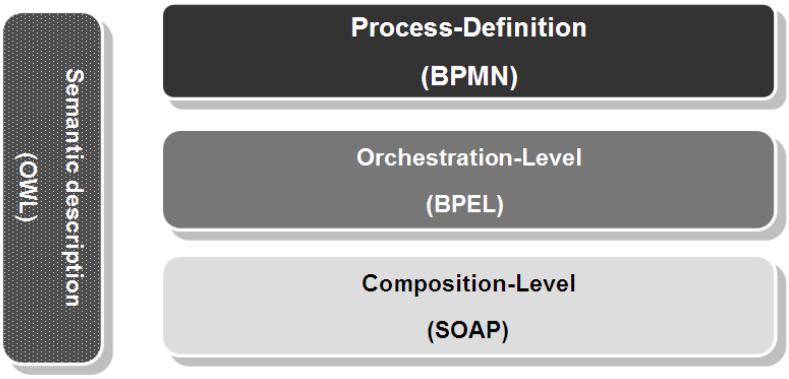


Figure 72: Level-based infrastructure composition

5.1. Process Definition

At first it is essential to describe the process model in a semantic manner to obtain a high-level view of the entire measurement process and to enforce a standard compliant procedure. Therefore, the Business Process Modeling Notation (BPMN) [OMG06] has been applied. In doing so this representation describes all processes, sub-processes, properties, and sub-properties of ISO/IEC 15939. This process model is used to divide the complete measurement process into different architectural components. Furthermore, the BPMN is used to produce the business process diagram on the basis of the ISO/IEC 15939 process model. Figure 73 presents the standard at a glance. Since he has been described in detail in chapter 2.3.3.3. The ISO/IEC15939 Software Measurement Process, the following explanations are shrieked to the BPMN implementation.

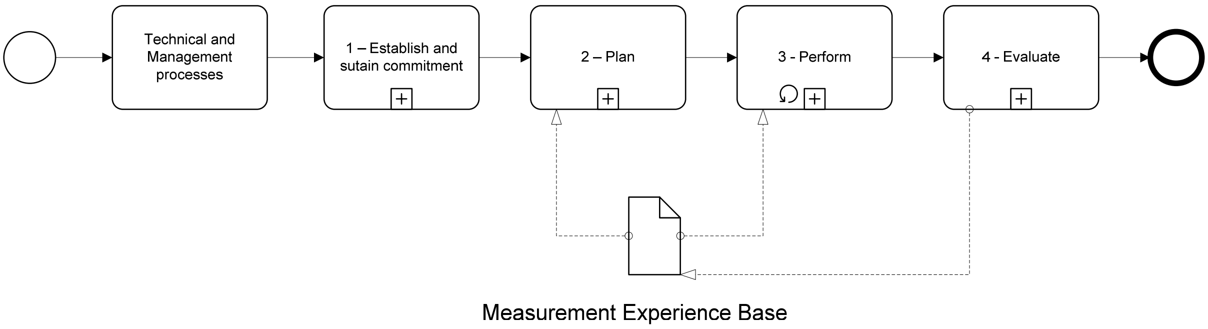

Technical and Management processes
1 – Establish and sutain commitment
2 – Plan
3 - Perform
4 - Evaluate
Measurement Experience Base

Figure 73: Simplified ISO/IEC 15939 process overview

Technical and Management Processes

The Technical and Management processes represent the overlying business processes where a requirement for measurement is driven by an information need which is intended to be satisfied by the result of an instance of a measurement process: information product.

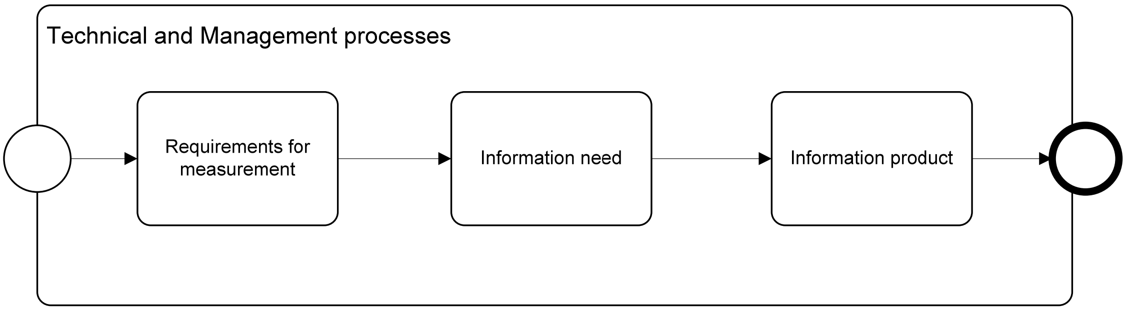

Technical and Management processes
Requirements for measurement
Information need
Information product

Figure 74: Technical and Management processes

The first process step is intended to establish and sustain measurement commitment. Assign resources and identify measurement scope are the subtask for this sub process.

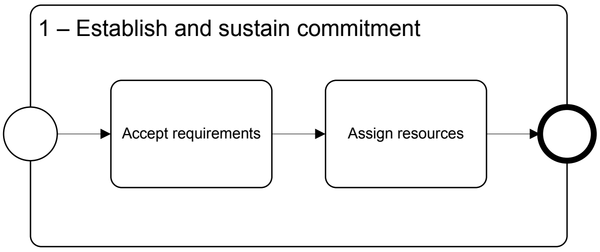


Figure 75: Establish and sustain commitment

The trigger for the measurement commitment has its origin in the *requirement for measurement* from Technical and Management processes.

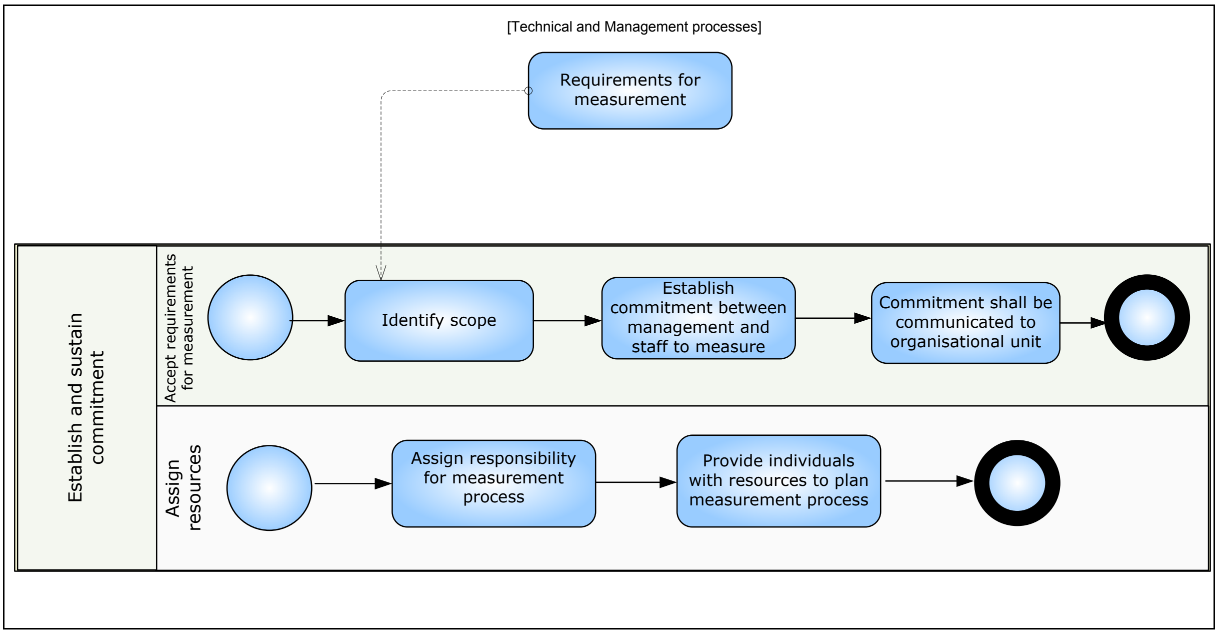

[Technical and Management processes]
Requirements for measurement
Establish and sustain commitment
Accept requirements for measurement
Identify scope
Establish commitment between management and staff to measure
Commitment shall be communicated to organisational unit
Assign resources
Assign responsibility for measurement process
Provide individuals with resources to plan measurement process

Figure 76: Detailed view on process Establish and sustain commitment

Plan the Measurement Process

The first out of two core measurement processes is *plan the measurement process.* Major input is logically the information need and the measurement commitment (scope and resource allocation). Additional improvement actions from earlier iterations of the measurement process

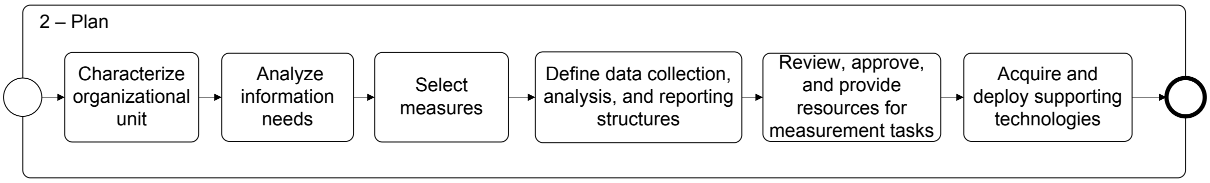

2 – Plan
Characterize organizational unit
Analyze information needs
Select measures
Define data collection, analysis, and reporting structures
Review, approve, and provide resources for measurement tasks
Acquire and deploy supporting technologies

Figure 77: Plan the Measurement process

The most important process step thereby is *analyze information needs* and within *select measures.* Activities in this regard should use experience from previous mappings of information needs and software measures and follow a standardized procedure (e.g. Goal-Question-Metric or Factor-Criteria-Metric paradigm). The second important step is the definition of evaluation criteria's. Thus, the definition of a quality model has to be done in the *plan* process.

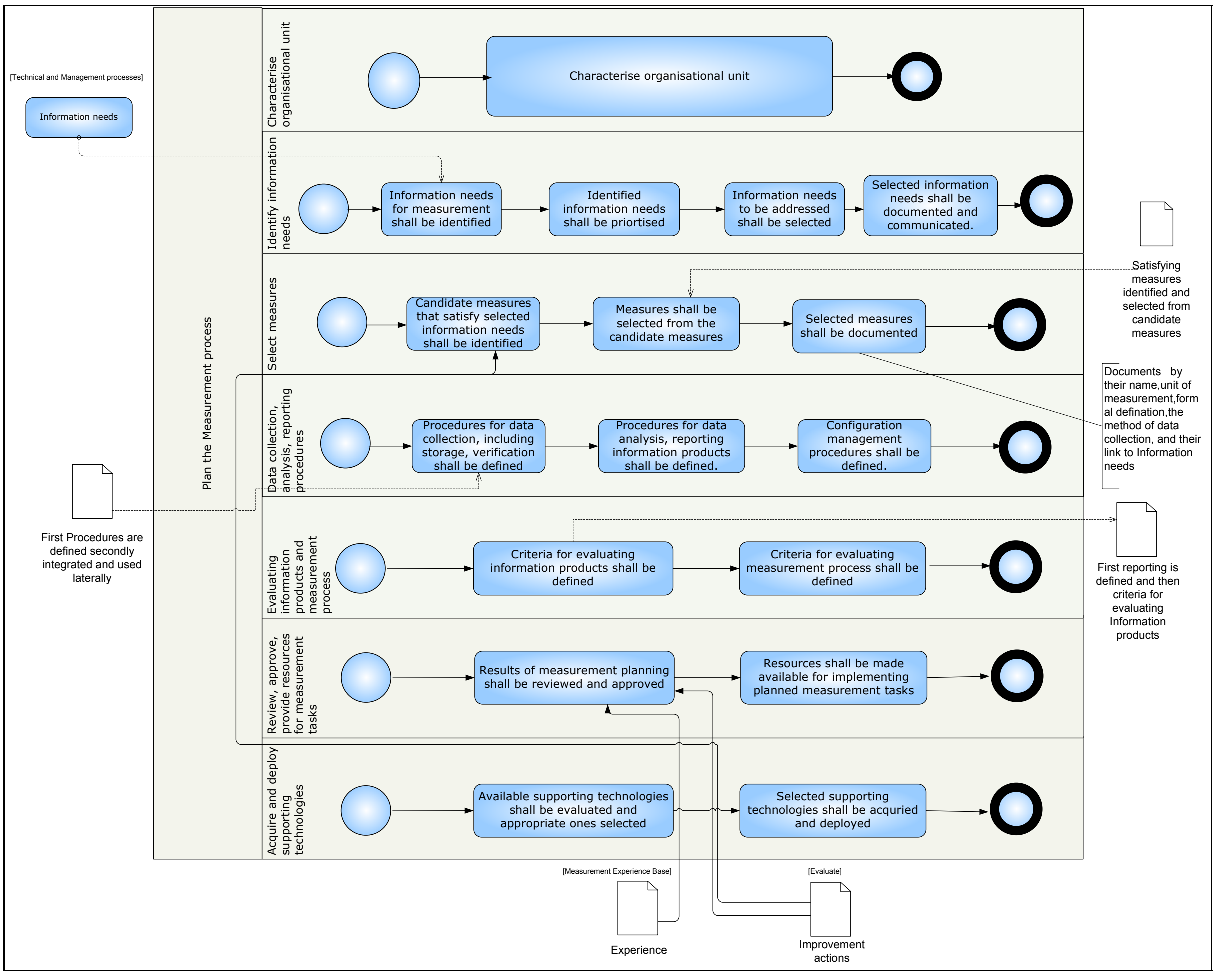

[Technical and Management processes]
Information needs
Plan the Measurement process
Characterise organisational unit
Characterise organisational unit
Identify information needs
Information needs for measurement shall be identified
Identified information needs shall be priortised
Information needs to be addressed shall be selected
Selected information needs shall be documented and communicated.
Select measures
Candidate measures that satisfy selected information needs shall be identified
Measures shall be selected from the candidate measures
Selected measures shall be documented
Satisfying measures identified and selected from candidate measures
Documents by their name,unit of measurement,formal defination,the method of data collection, and their link to Information needs
Data collection, analysis, reporting procedures
Procedures for data collection, including storage, verification shall be defined
Procedures for data analysis, reporting information products shall be defined.
Configuration management procedures shall be defined.
First Procedures are defined secondly integrated and used laterally
Evaluating information products and measurement process
Criteria for evaluating information products shall be defined
Criteria for evaluating measurement process shall be defined
First reporting is defined and then criteria for evaluating Information products
Review, approve, provide resources for measurement tasks
Results of measurement planning shall be reviewed and approved
Resources shall be made available for implementing planned measurement tasks
Acquire and deploy supporting technologies
Available supporting technologies shall be evaluated and appropriate ones selected
Selected supporting technologies shall be acquried and deployed
[Measurement Experience Base]
Experience
[Evaluate]
Improvement actions

Figure 78: Detailed view on Plan the Measurement process

The results of the sub process are a set of appropriate measures aligned with project specific threshold as a quality model.

Perform the Measurement Process

The second core measurement process is *Perform the measurement process*. Thereby the measurement of software engineering artifacts, data collection and analysis, as well as the communication of results is the major steps. The major output of the *perform process* and therefore of the core measurement process is the information product.

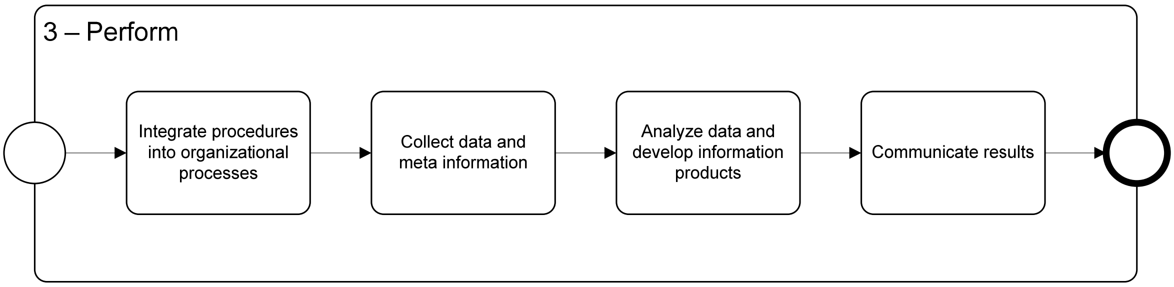

3 – Perform
Integrate procedures into organizational processes
Collect data and meta information
Analyze data and develop information products
Communicate results

Figure 79: Perform the Measurement Process

Experience from previous iteration is mainly used in regards of measurement analysis. The analysis and results communication will be done in an iterative manner as long as measurement data is collected.

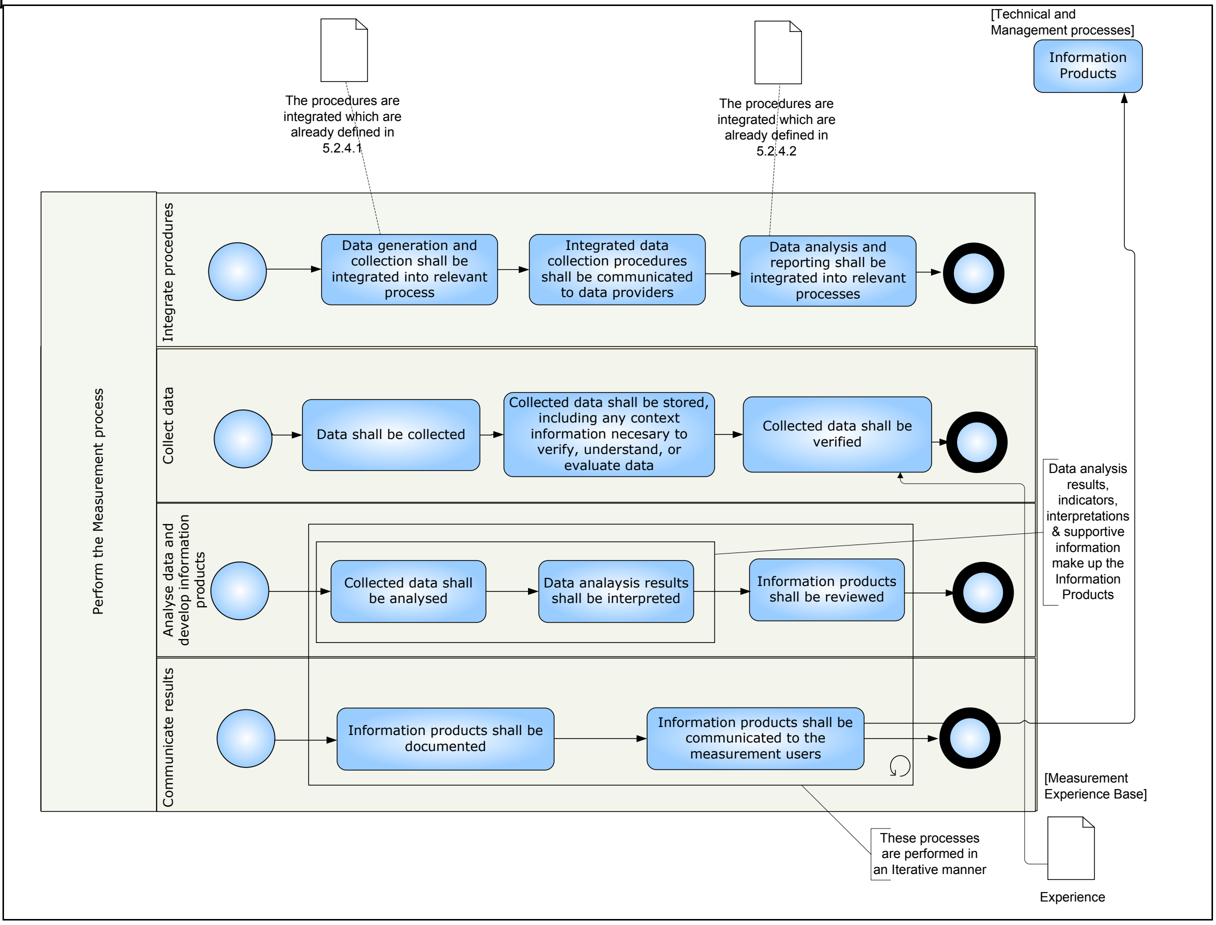

[Technical and Management processes]
Information Products
The procedures are integrated which are already defined in 5.2.4.1
The procedures are integrated which are already defined in 5.2.4.2
Perform the Measurement process
Integrate procedures
Data generation and collection shall be integrated into relevant process
Integrated data collection procedures shall be communicated to data providers
Data analysis and reporting shall be integrated into relevant processes
Collect data
Data shall be collected
Collected data shall be stored, including any context information necesary to verify, understand, or evaluate data
Collected data shall be verified
Analyse data and develop information products
Collected data shall be analysed
Data analaysis results shall be interpreted
Information products shall be reviewed
Data analysis results, indicators, interpretations & supportive information make up the Information Products
Communicate results
Information products shall be documented
Information products shall be communicated to the measurement users
[Measurement Experience Base]
These processes are performed in an Iterative manner
Experience

Figure 80: Detailed view on Perform the Measurement process

Evaluate Measurement

The last process step and the most unique one of the ISO/IEC 15939 measurement process is *Evaluate measurement.* As the name implies the focus is to evaluate previous procedures and to identify potential improvements. Especially the mapping of information need and measure should be evaluated in this process step.

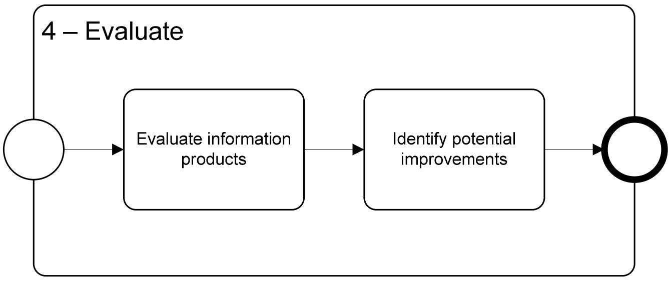

4 – Evaluate
Evaluate information products
Identify potential improvements

Figure 81: Evaluate measurement

Beside the pure software measurement data (derived from *perform)* the measurement experience base is feed with information in this sub process.

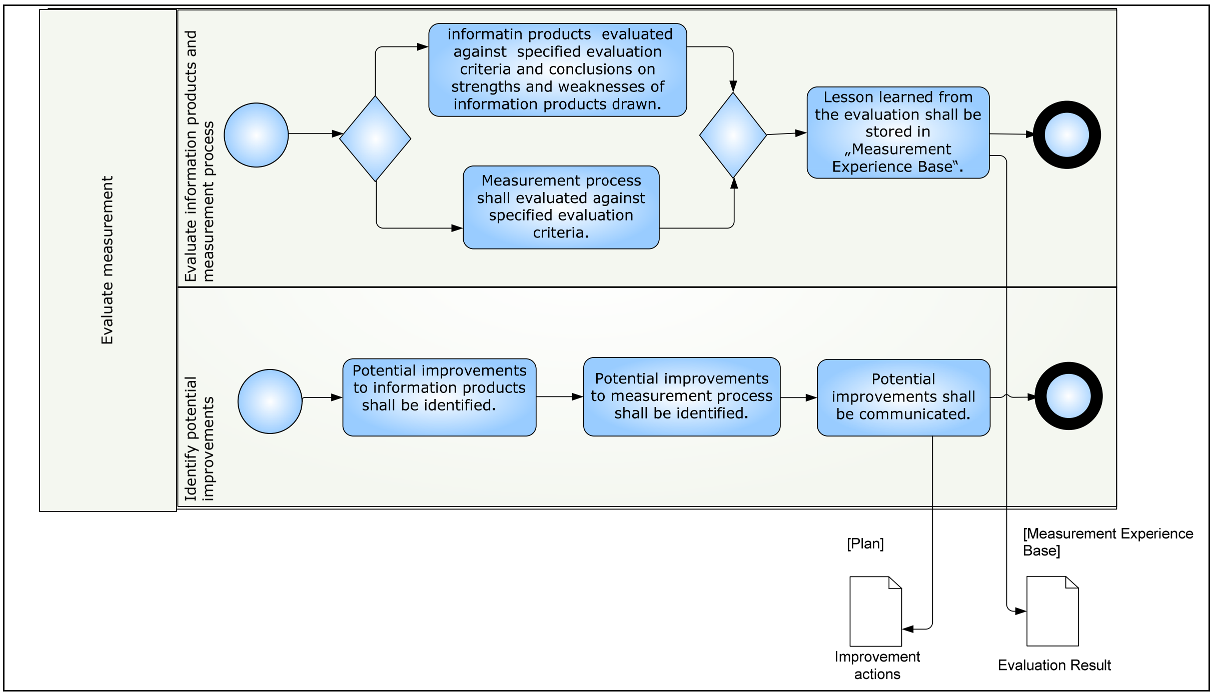

Evaluate measurement
Evaluate information products and measurement process
informatin products evaluated against specified evaluation criteria and conclusions on strengths and weaknesses of information products drawn.
Measurement process shall evaluated against specified evaluation criteria.
Lesson learned from the evaluation shall be stored in „Measurement Experience Base".
Identify potential improvements
Potential improvements to information products shall be identified.
Potential improvements to measurement process shall be identified.
Potential improvements shall be communicated.
[Plan]
[Measurement Experience Base]
Improvement actions
Evaluation Result

Figure 82: Detailed view on Evaluate measurement

5.2. Service-oriented Measurement Infrastructure detailed Description

The automation and technical support of the several processes and sub processes differs between each other. The fulfillment of the measurement of identified measures and the collection and storage of the measurement data is freely offered by existing tools but as described in previous chapters a set of different services is necessary.

In general a service-oriented measurement infrastructure need a throughout support of a complete measurement process. That drives the look to sub processes which are barely automated or supported: the mapping of an information need with distinct measure and the evaluation of measurement processes. But obviously this throughout implementation is necessary for a proof of concept of a service-oriented measurement infrastructure.

The question arise which use case is appropriate for a first prototype, which components from common SOA's could be used and which components have to be implemented specific as measurement services.

Some restrictions have been made regarding supported measures and measurement artifacts. To choose a widespread and well accepted set of measures for a relevant use case the Chidamber & Kemerer metrics for object-oriented design have been selected. The measurable artifacts have been constricted to Java as the major programming language in the scientific area as well as the open source community. But even with these constraints the pictured scenario (C&K metrics and Java technology) can be taken as widespread in software engineering practice.

Since, their exists manifold possible technical solutions for every described framework element and the fact that the technical solutions are only part of a device record out of which a distinct infrastructure is created at runtime. The presented implementations can only provide a glimpse of resulting infrastructures.

5.2.1. GQM Process Model

For the task of mapping the information need with a measure two standardized approaches has been applied: the Goal-Question-Metric Method (GQM) and the machine readable description of knowledge about metrics with ontology's.

As presented in chapter 2.3.3.1. The Goal-Question-Metric Method GQM contains a holistic measurement process. Since, the ISO/IEC 15939 was chosen as the overall process model, the following BPMN diagram is limited to two sub processes representing the GQM paradigm for selecting appropriate measures for defined goals and thereby information need.

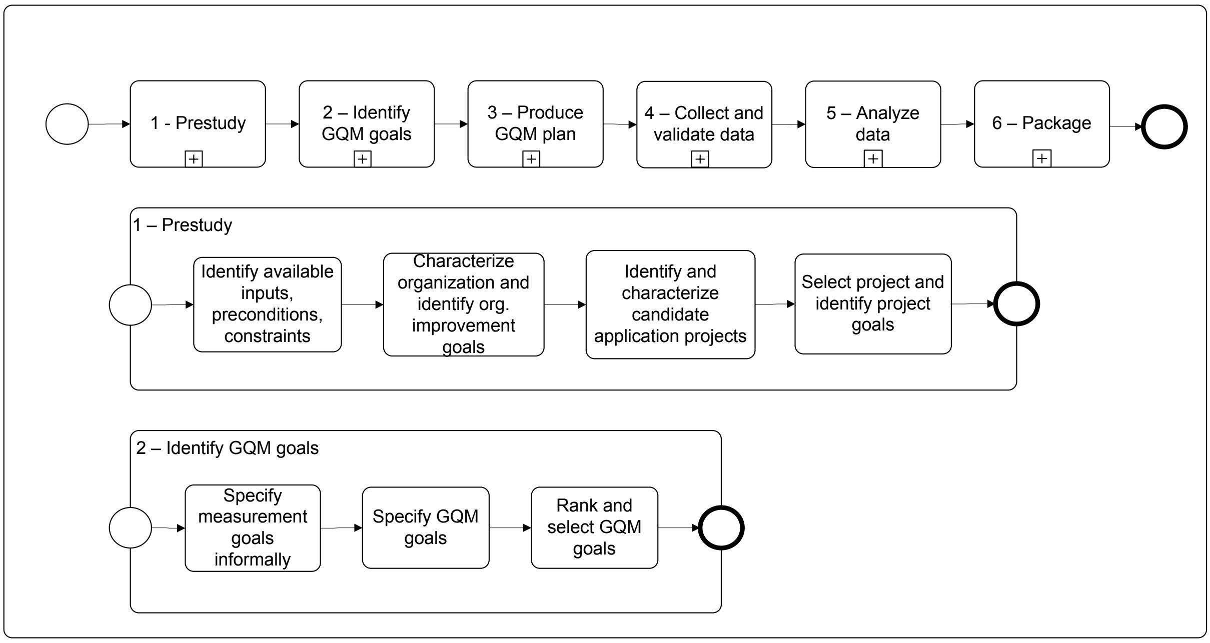

1 - Prestudy
2 – Identify GQM goals
3 – Produce GQM plan
4 – Collect and validate data
5 – Analyze data
6 – Package
1 – Prestudy
Identify available inputs, preconditions, constraints
Characterize organization and identify org. improvement goals
Identify and characterize candidate application projects
Select project and identify project goals
2 – Identify GQM goals
Specify measurement goals informally
Specify GQM goals
Rank and select GQM goals

Figure 83: BPMN diagram of GQM paradigm

5.2.2. Ontology for Object-oriented Metrics

Ontology's are a fundamental concept of the Semantic Web envisioned by Tim Berners-Lee [LeeHendler+01]. Together with explicit representation of the semantics of data for machine-accessibility, such domain theories are the basis for intelligent next generation applications for the web [AngMarOls+03] and other areas of interest [Devedzic06] with a special focus on knowledge sharing and reuse. Ontology's are also basis for interaction and work of different agents or applications [KernchenRud+07] [MenckeKunz+08a]. Top-level application areas identified by [Fikes98] are collaboration, interoperation, education, and modeling.

Ontology's can be defined as a specification of a conceptualization [Gruber93], or in other words as the formal representation of an abstract view of the world. They include a vocabulary, instances, taxonomy, relations, and axioms about a certain domain.

A vocabulary defines terms with unambiguous meanings. Furthermore, logical statements for the description of terms and rules for their combination and relation are provided. A taxonomy is part of the ontology concept for a hierarchical classification in a machine-process able form. Individuals/instances represent the objects of the ontology and thereby the available knowledge, while classes/concepts describe abstract sets of individuals. Attributes can be assigned to instances for description. They have a name and value. The last key concept of ontology's is the relation. It can be described by using attributes and assigning another individual as a value. Common relation types are the *is-a* relation (subsumption relation) and the *part-of* relation (metonymy relation). The possibility to define special domain specific relations is a considerable additional value of the concept of an ontology. Axioms are always true and represent knowledge that is not inferable from other individuals.

It is possible to distinguish ontology's in two broad categories: lightweight and heavyweight ontology's. A lightweight ontology is described by individuals, classes, attributes, relations and axioms, meanwhile heavyweight ontology's are an extension of lightweight ones by the additional usage of axioms for a more detailed domain description [MenckeKunz$^+$08b].

There already exist ontology's for different fields of application. Some are available via libraries like the DAML ontology library [DAML08] and the SchemaWeb library [Schema08].

The first part of the ontology connects a distinct information need with a software characteristic. The software characteristics can be distinguished into product, process, or resource related. For resources the ontology defines among others costs, experience, and magnitude, for process among others time and effort. For the product characteristics artifacts like size and complexity were taken into account as well as aspects of the ISO/IEC 9126 standard for product quality. Some standardized examples are efficiency, portability, reusability, design, reliability, usability, and availability. Considering the general goal of object oriented metrics it is clear that product quality and in this case the design is the main focus for object-oriented software characteristics. The other elements have been defined to provide an extensible model [Kunz$^+$06a].

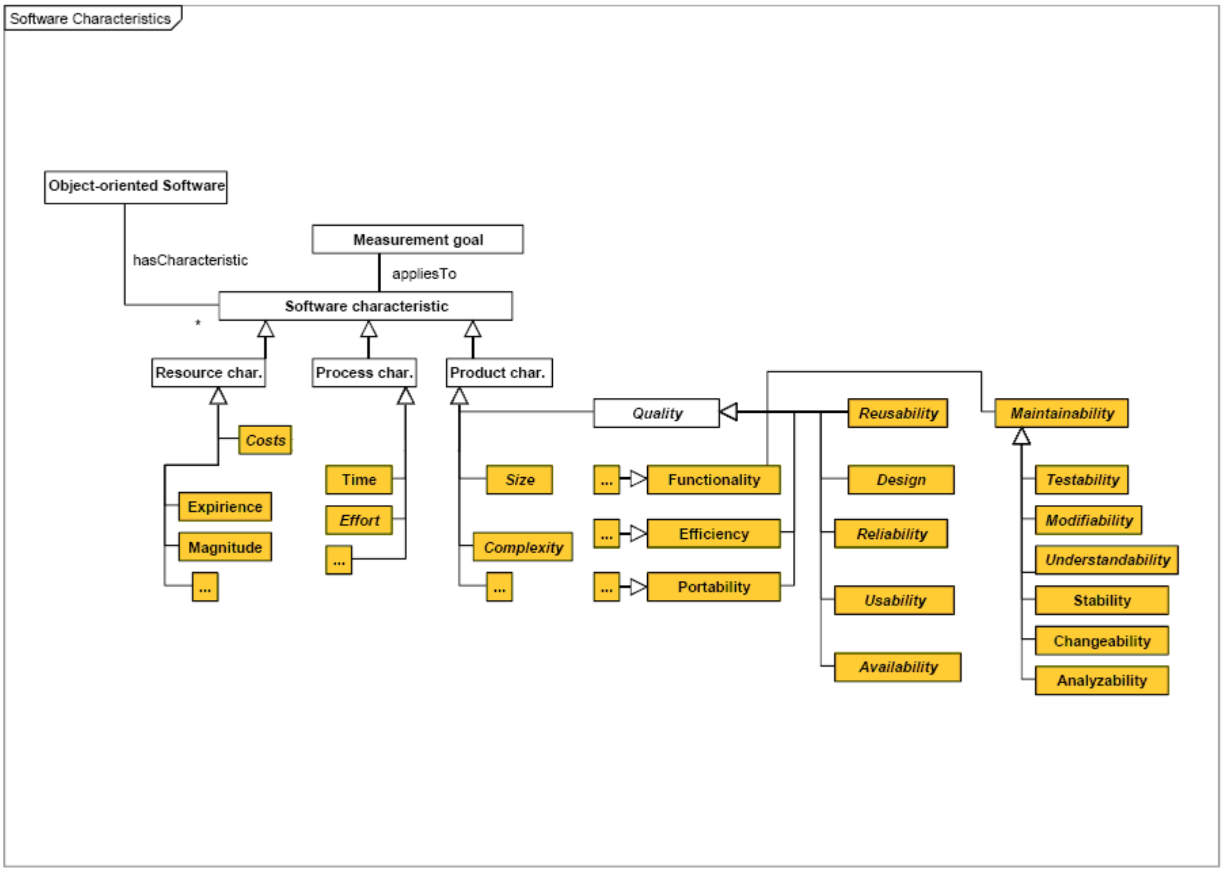


Figure 84: Ontology component "Software Characteristic" [Weise06]

Another important part of the ontology is the consideration of object-oriented structure and concept. The logical structure of object-oriented objects like encapsulation or coupling is often a measurable artifact. Therefore the measurable artifact is divided into object-oriented structure and object-oriented concept.

The object-oriented structure is arranged by using different elements with correlated relations.

Another important feature in this ontology component is the *"usesElement"* association. Thereby a *"using"*-relation between two elements is described. A metric measures an attribute of an element by using another element. For example the size of a class (*"isDefinedFor"*) is measured by counting its included methods (*"usesElement"*).

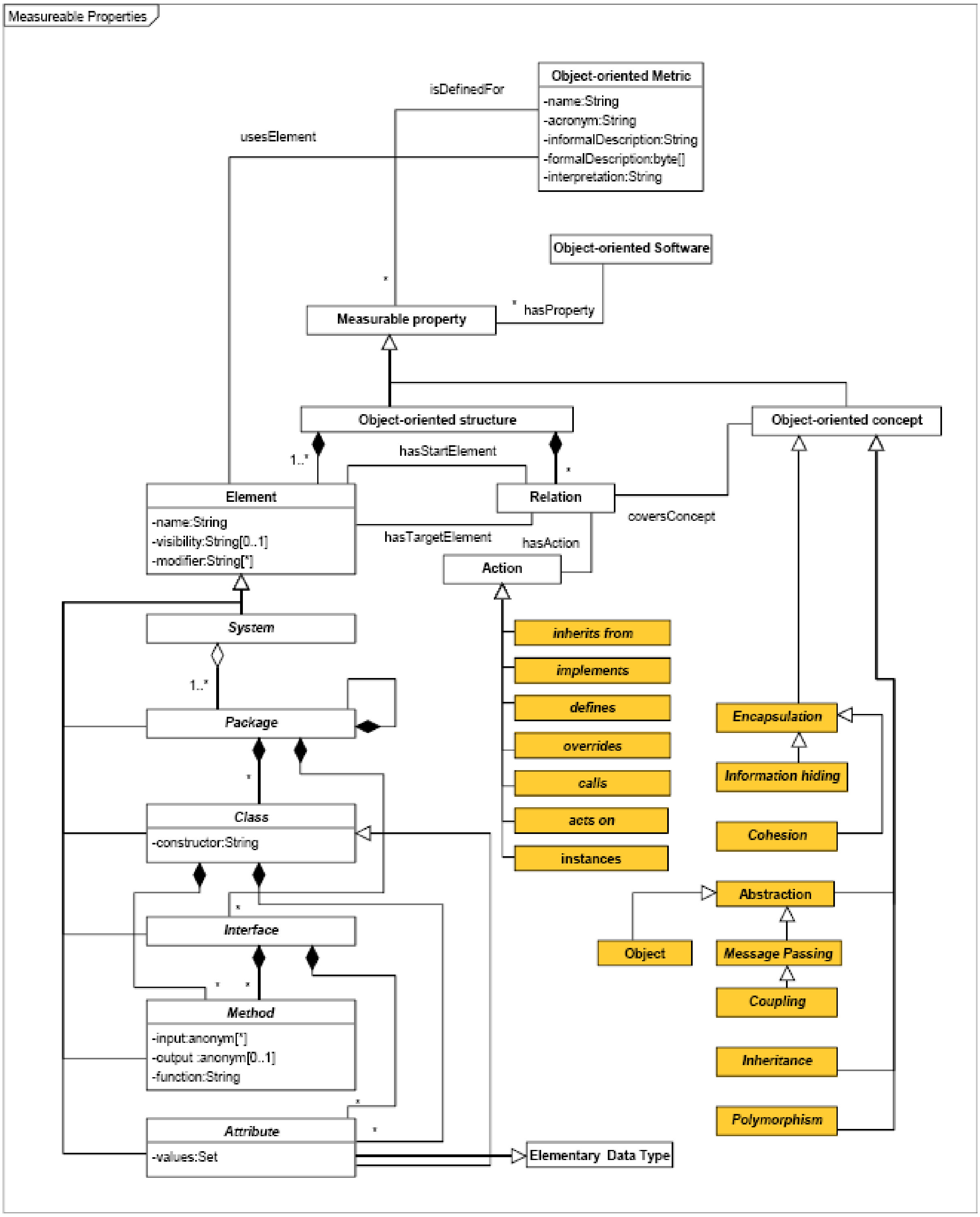


Figure 85: Ontology component “Measurable Properties”

As previously described the target of the ontology is to connect a distinct information need, which must be satisfiable by measurements, with a specific measure. The need itself evolves from requirements for control and optimization of software and related processes and resources.

Based on an analysis of the problem area a concept model is created. It aims to connect object-oriented metrics with a measurement goal which refers to distinct software characteristics with predications which are based on measurement results. Furthermore evaluation and interpretation criteria are included.

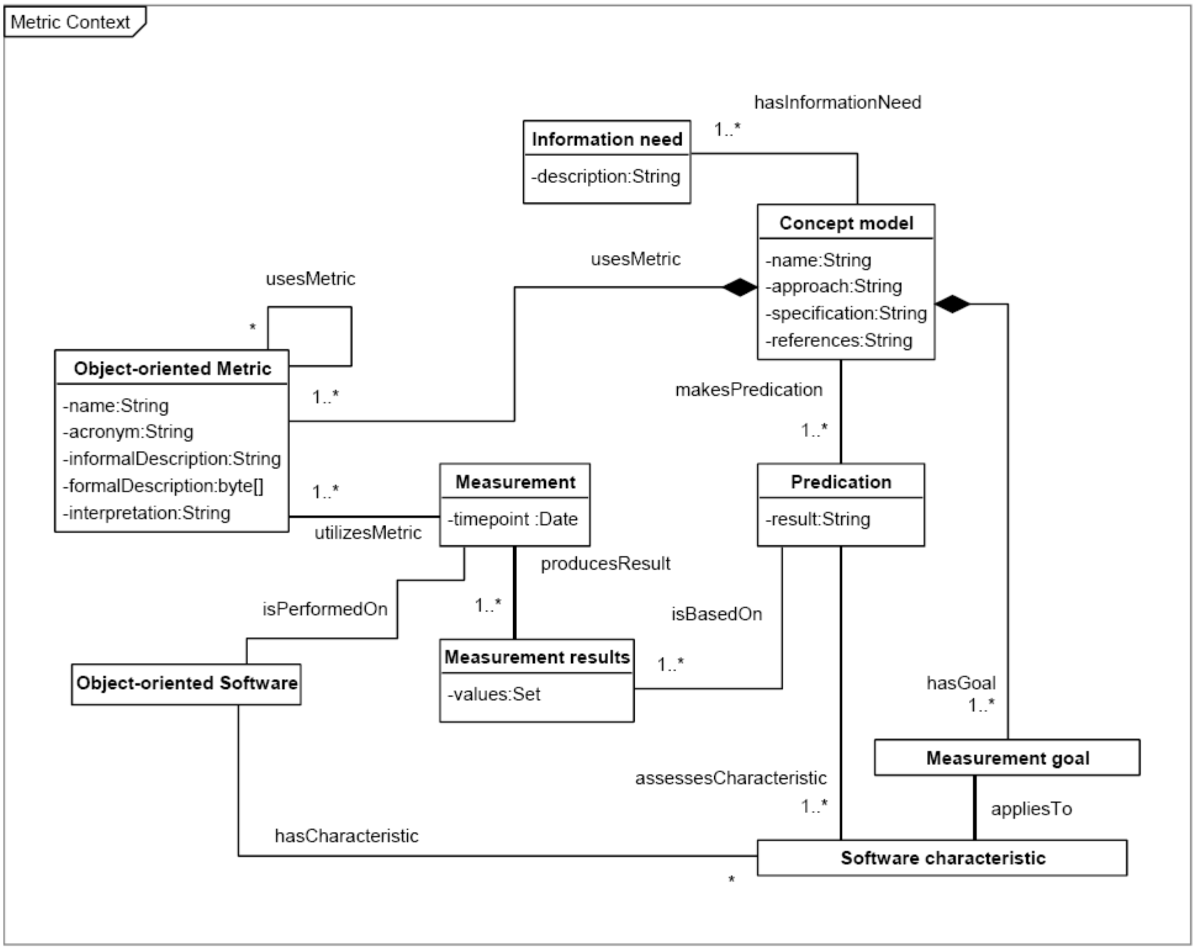


Figure 86: Ontology component "Metric Context"

The metrics themselves can be distinguished in base and derived metrics and they can be classified into product, process or resource metric.

In a similar way a metric is defined for a measurable property which connects it to an object-oriented structure respectively an object-oriented concept.

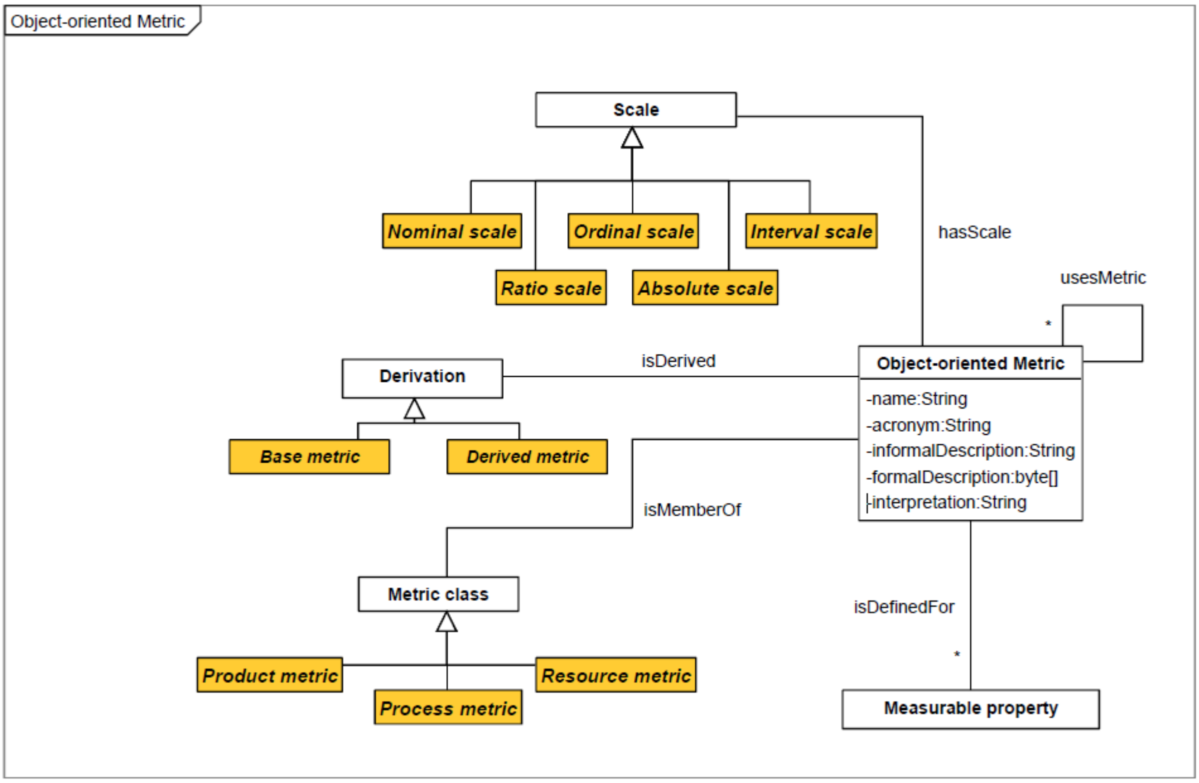


Figure 87: Ontology component "Object-oriented Metric"

In the following figure all components are arranged in the intended manner as Object-Oriented Metrics Ontology (OOMO).

An information need is connected with a concept model and in this way with a measurement goal and with a distinct software characteristic.

The software characteristic is connected with a measurable property which is defined for an object-oriented metric and the according metrics attributes.

Figure 88: **Ontology for software measures** presents the interplay of the aforementioned components into the ontology for object-oriented metrics.

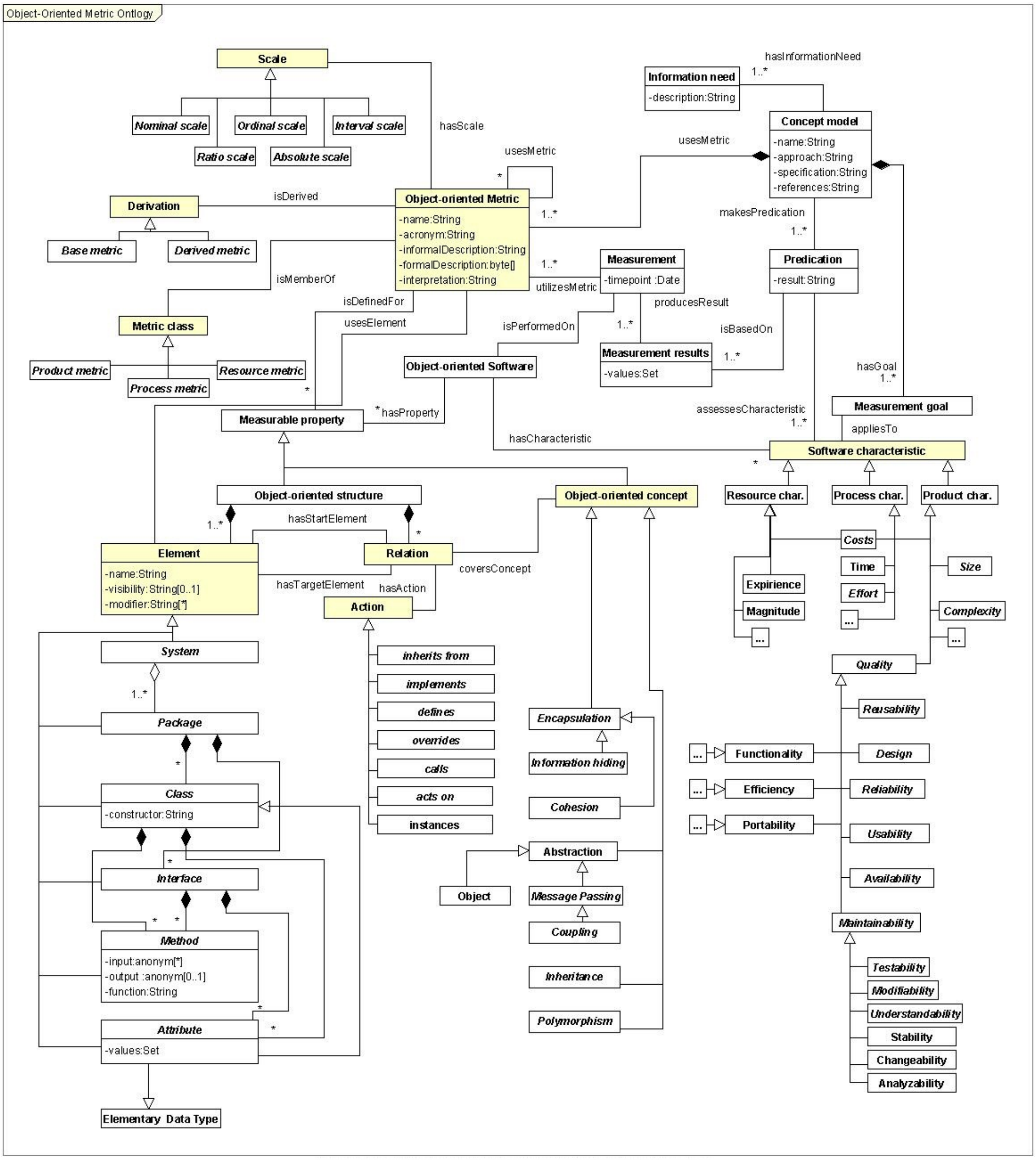
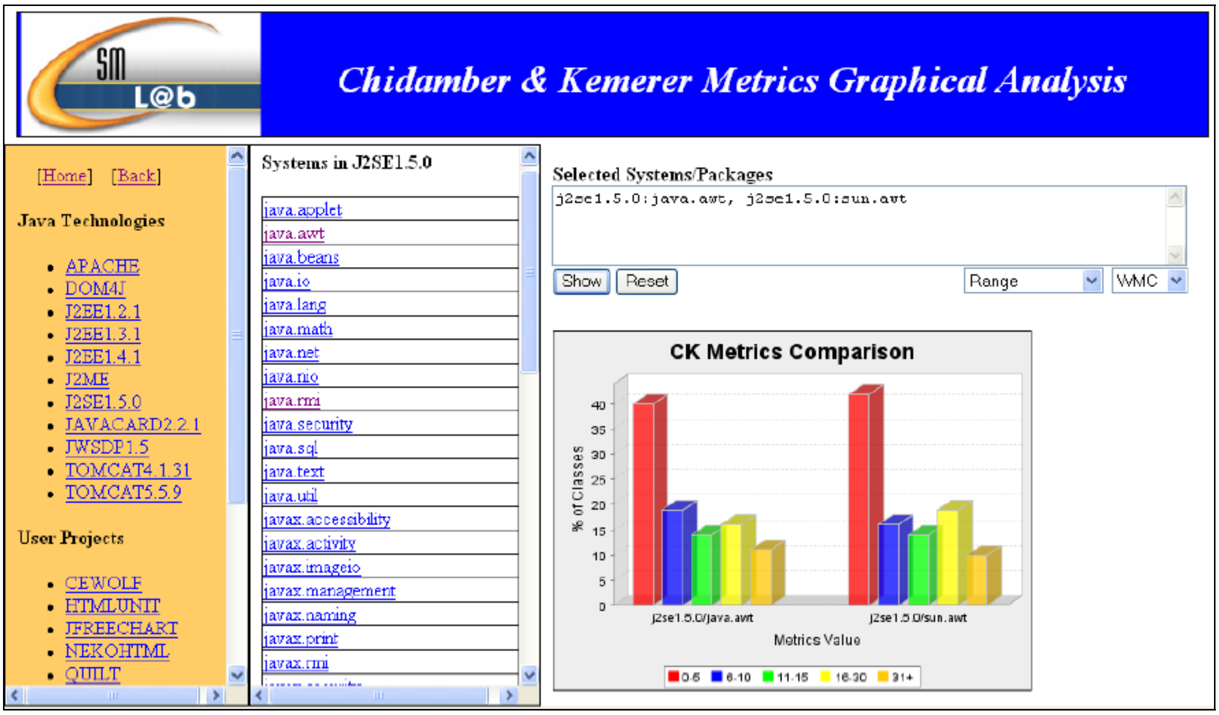


Figure 88: Ontology for software measures

5.2.3. Web Service for Object-oriented metrics

Another core component for a service-oriented measurement infrastructure is for sure a web service for performing measurement activities. This task has been realized as part of a master thesis project at the SML@b [Farooq05]. The implementation of the Chidamber&Kemerer metrics for object-oriented design as a web service creates an important building block for the proof-of-concept of the service-oriented measurement infrastructure. By including the results of the evaluation of a comprehensive set of open-source libraries and applications the web service creates the groundwork for an extensive set of empirical data. This empirical data can be used for benchmarking user projects against the industry standard represented by a large set of open-source applications [FarKerKunz+06] [BraFarKunz+06] (A detailed description of the collection and storage of empirical data is shown in chapter 5.5. Service-oriented Measurement Database).



Chidamber & Kemerer Metrics Graphical Analysis

Selected Systems/Packages

j2se1.5.0:java.awt, j2se1.5.0:sun.awt

Figure 89: Web service for measuring C&K metrics [Farooq05]

The GUI has been created as a prototype and is not part of the final infrastructure. But it enables and idea of analysis possibilities provided by the measured data. Figure 89 shows a screenshot of the implemented measurement analysis. In this case a comparison of two Java software systems.

A detailed description how visualization is realizes in the intended infrastructure can be found in chapter 5.4. Graphical User Interface

For the integration of this service into the SOMI it is more important how the measurement results are provided for other services beside the graphical visualization. To provide a feasible interface the measurement results are captures as an XML file. Figure 90 shows an example for a measurement result containing the different measures.

```
<?xml version="1.0" encoding="UTF-8"?>

<!—C&K Metrics Results for the dom4j technology (code author:User)-->

-<dom4j>

<!-- org.dom4j.dom element contains metrics results for DOMText packages.-

       -<dom4j-1.6.1>

              <NOC>0</NOC>

              <LOC>64</LOC>

              <DIT>5</DIT>

              <WMC>38</WMC>

              <CBO>7</CBO>

              <RFC>47</RFC>

              <LCOM>703</LCOM>

                                   </dom4j-1.6.1>

-</dom4j>
```

Figure 90: Example for a XML result file for a measurement

The interaction of the different actors and web service elements regarding measurement and data retrieving are shown in the sequence diagrams (Figure 91, Figure 93) and the different activities are presented in the activity diagram (Figure 92).

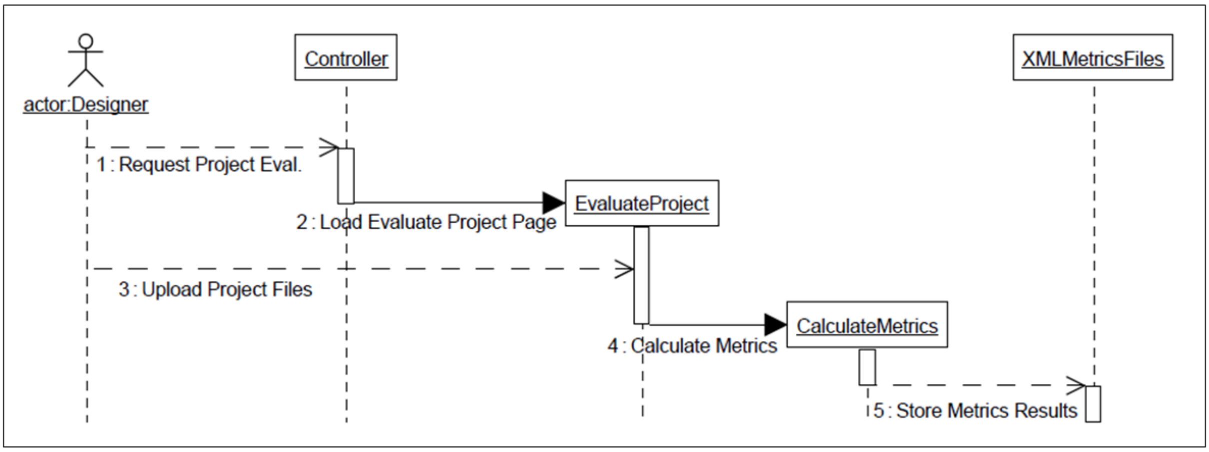


Figure 91: Sequence diagram for measuring a Java project [Farooq05]

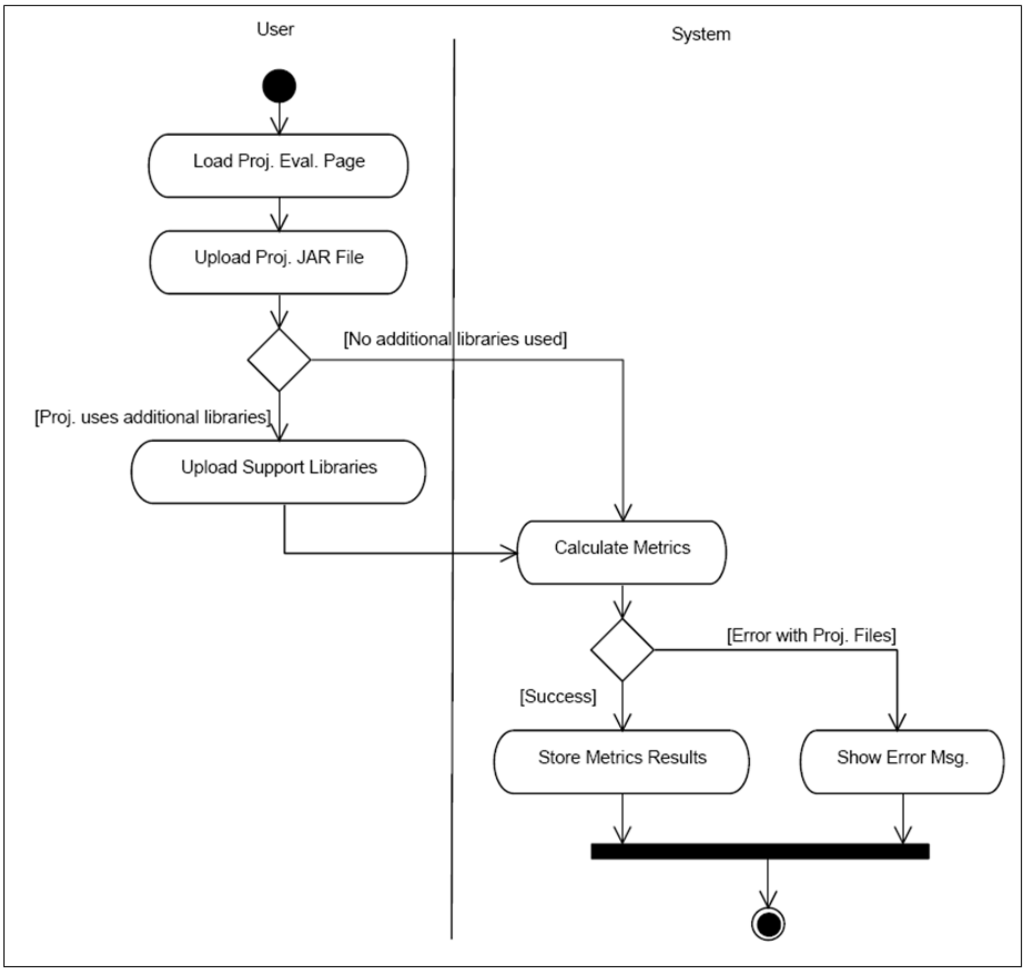


Figure 92: Activity diagram for measuring a Java project

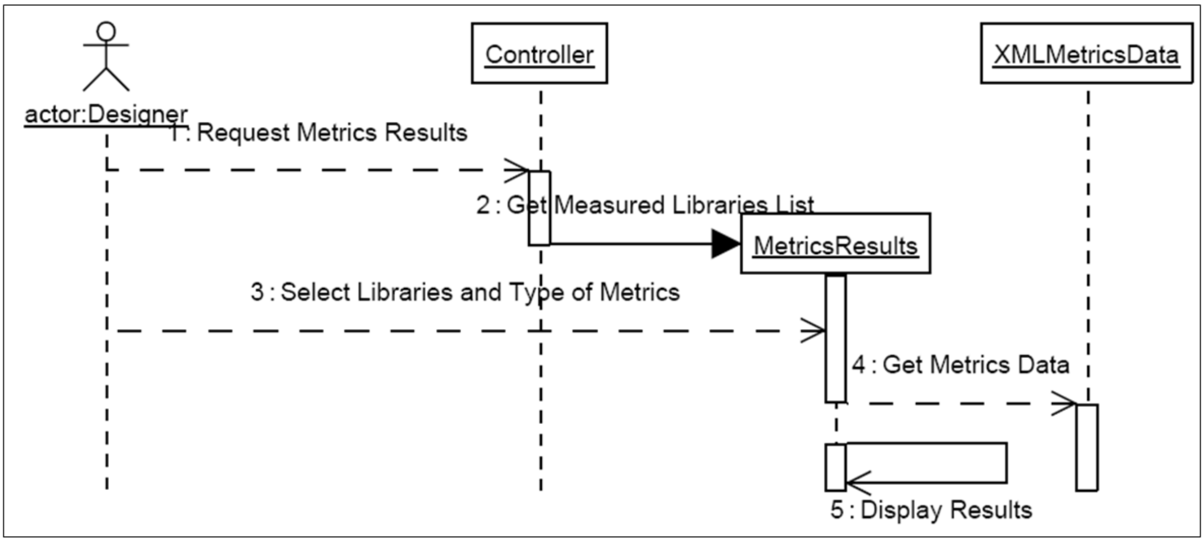

actor:Designer
Controller
XMLMetricsData
1 : Request Metrics Results
2 : Get Measured Libraries List
MetricsResults
3 : Select Libraries and Type of Metrics
4 : Get Metrics Data
5 : Display Results

Figure 93: Sequence diagram for measurement data retrieving

5.2.4. Search and Integration Process for Measurement Services

To realize the components for the service directory and measurement service integration we choose the approach of a SOA-Service-Center [SchmiDim04]. Such a service center supports different tasks and implies the use of already existing tools and new approaches. It provides several functionalities like a directory about all available services, descriptions for provided interfaces, and information about existing service compositions. A second important information in this concrete use case aims at the number and types of metrics that a distinct measurement service supports. In addition, one can add Meta information about the provided services in such a service center, for example sequence relationships or expected quality behavior.

But more important for building up an infrastructure are the provided search functionalities. Therefore, the service center offers functionalities like runtime search, criteria-based search, and information where the services already have been used [SchmiDim04].

The second important question addresses the security of measurement services. Logically, no user will accept that his measurement objects become public. Therefore, it is necessary that the integration architecture is able to ensure that the web services which are being used to build the measurement infrastructure are compliant to the security policy of the concrete environment.

On this account the Web Service Security Framework has been used to describe how to incorporate security technologies (e.g. Kerberos [MIT05]) into web services by defining a place for them within the SOAP headers [NewLom03]. The WS-Security specification targets message alteration by including digital signatures and message disclosure by supporting message encryption, while it can also be used to authenticate messages through the use of Kerberos [MIT05].

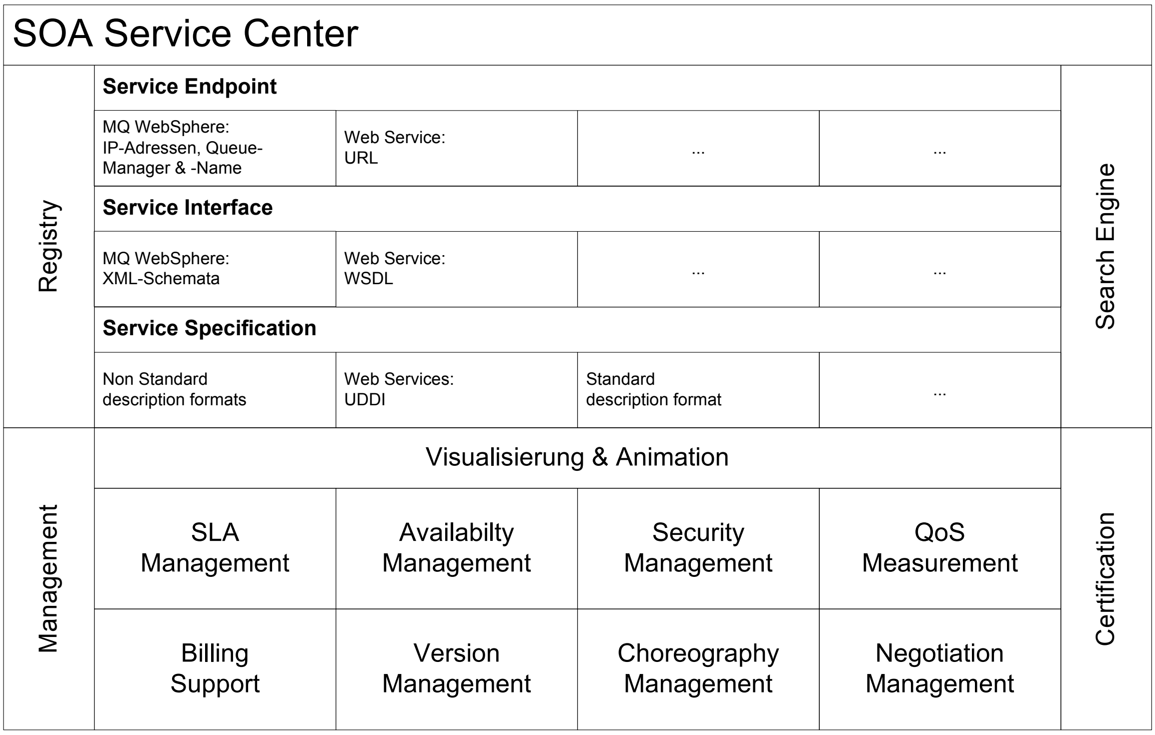


Figure 94: SOA Service Center [SchmiDim04]

This SOA Service Center is not limited to measurement infrastructures nor was it developed particular for this field of application. But this Service Center is a good example that using the SOA technology enables the integration of new technologies in the area of software measurement.

Even if some of the components contain a own GUI for demonstration purpose the general goal is to create a complete infrastructure by the use of SOA and web services with the aim to be viewed as one software measurement tool. That establishes a need for a uniform interaction with a user.

Referring back to the foundations of software measurement it was mentioned those different users are stakeholder in a software measurement program.

On the one hand for a software developer the measurement process should disappear and on the other hand capabilities are needed for quality engineers if measurement indicates lack of quality.

5.3. Quality driven Assembly of Web Services

The importance of providing service-oriented architectures in every field of application is beyond controversy these days and applied in high diversity in different economic fields. Unfortunately existing solutions are focusing mainly on the provided functionality. But for the success of Systems Integration in the long run, the quality of developed architectures is of substantial interest [Kunz^{+}08a] [Kunz^{+}08b].

Existing approaches relates to QoS (Quality of Service) parameters such as throughput, response time and availability. Similar approaches are introduced by Menascé and Dubey [MenDub07] or Sirin, Parsia and Hendler [SirinParsia+05]. But these QoS parameters describe only single aspects out of many and the orientation on network related aspects like performance, availability, capacity, integrity, security, and accessibility leads to the result that a lot of services lack on other quality facets like reusability, portability, or maintainability.

In a Gartner study the adverse situation in the usage of services was summarized by the quotation [Chappell06] [Gartner07]:

"You should expect to reuse only a fraction of your services, maybe just 20% of them."

Probably it is not astonishing that reusability is not being reached if it is not measured previously.

For the enhancement of QoS attributes with comprehensive product quality attributes a framework for quality-driven creation of architectures is proposed in this chapter. Besides these quality-oriented characteristic the usage of semantic knowledge and structured process descriptions enable an automatic procedure. Especially the combination of both is a promising approach. Due to manifold advantages of high-flexible infrastructures compared to monolithic products a lot of initiatives propose approaches for the integration of single components (e.g. services). Semantic metadata provides the basis for the automation of this process. But those approaches lack from a throughout consideration of empirical data. Either only functional requirements or single quality attributes are taken into consideration.

In contrast to existing approaches the presented framework reveals a holistic orientation on quality aspects. It combines semantic web technologies for the fast and correct assembly of system elements and quality attribute evaluations for making the best assembly decisions possible. Therefore complex quality models [AbrKunz+03] are considered as well as empirical evaluations. Furthermore different types of quality evaluation like simulation and static and dynamic software measurement are used. Combining them delivers a holistic quality view on components and the flexibility enables a quality improvement of the targeted system by the exchange of single components if the evaluation of their quality attributes decreases.

The presented general QuaD^2-Framework (Quality Driven Design) can easily be adapted to many different fields of application, e.g. service-oriented architectures or enterprise application integration [KerKunz+07a]. In this way the framework can be a meaningful approach for service-oriented architectures in a general way beside the service-oriented measurement infrastructure [Kunz+08c] [MenckeKunz+08d].

5.3.1. QuaD^2 Framework

In general the sub processes of this empirical-based assembly process are the initialization, the feasibility check (checking the functional coverage), and the selection process based on empiricism as well as the operation of the established application. Quality assurance is achieved by certain sub processes that allow optimizations at initialization time as well as during runtime. Furthermore measurement sub processes are performed to update evaluation data.

The major goal of the described core process is an architecture consisting of single services. Such a service contains metadata-annotated functionality.

In order to achieve the sketched goals a special process is developed below. Its major use cases are introduced in Figure 95.

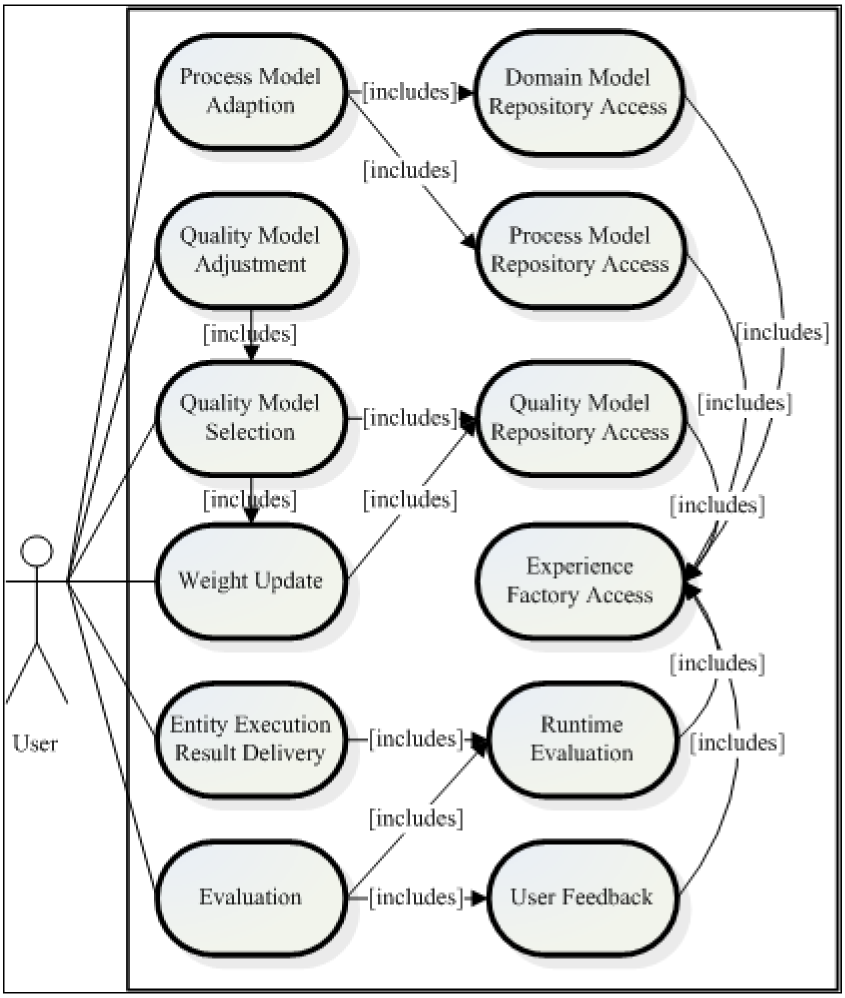

User
Process Model Adaption
Domain Model Repository Access
Quality Model Adjustment
Process Model Repository Access
Quality Model Selection
Quality Model Repository Access
Weight Update
Experience Factory Access
Entity Execution Result Delivery
Runtime Evaluation
Evaluation
User Feedback
[includes]

Figure 95: Use Case Diagram: Empirical-Based Service Orchestration Process

The basis of the presented approach is a collection of semantically-annotated sources: the process model repository, the service repository, a quality model repository and furthermore an experience factory.

The process model repository is the source for process models that serves as description for the functionality of the aspired distributed system. Example for such processes can be ISO/IEC 15939 [ISO/IEC02] for the software measurement process or didactical approaches [Mencke08]. Technological realization may vary, too. They can result in UML [OMG07], BPMN [OMG05], ontology's [Mencke07], etc.

An important source for empirical quality evaluations are quality models being provided by a quality model repository. The basis of a quality model's definition is an extensible list of quality attributes. The specification of a certain quality model is realized by selecting and weighting appropriate attributes. The evaluation and selection of appropriate services is based on evaluation criteria for each included attribute. Such attributes can be e.g. cost, performance, availability, security, and usability. The attributes and corresponding evaluation formulas are standardized e.g. in ISO/IEC 9126 [ISO/IEC01].

The service repository contains services, their semantic description and their evaluation data regarding all defined quality attributes.

The selection and adoption of process models and quality models are difficult tasks which constitutes the need for guidance and support. Because of this, the presented framework proposes the usage of existing experiences and knowledge about previously defined and used process models and quality models to support both process steps. Based on the Quality Improvement Paradigm, Basili and Rombach proposed the usage of an Experience Factory which contains among others an Experience Base and Lessons Learned [Basili94], [Basili99].

In the presented framework, the Experience Factory is fed from the process evaluation process and is the major building block to save empirical data and the user's experiences with specific process procedures or with distinct quality attributes.

Figure 96 defines the used diagram elements for the diagrams below; optional elements have a grey border.

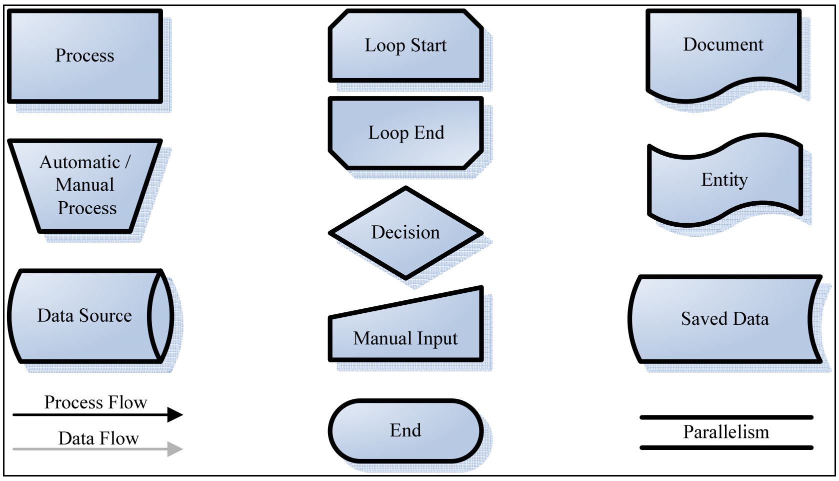


Figure 96: Definition of used diagram elements

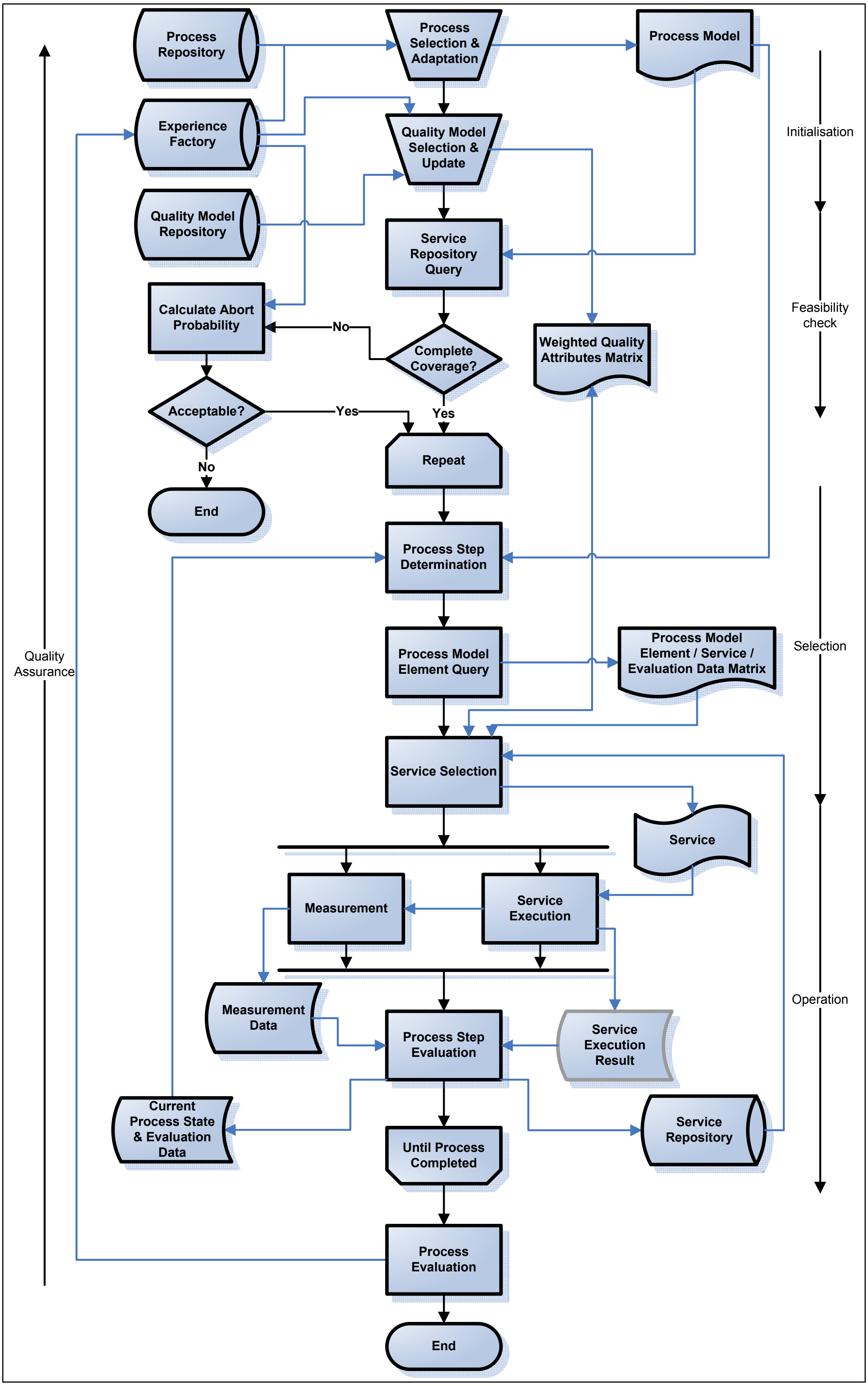


Figure 97: Quad^2 Framework

The focus on quality is a throughout property of the developed process and results in certain measurement and evaluation sub processes that are introduced in the following general process description and are described more detailed in subsequent sections. The derived results are directly used for optimization purposes [MenckeKunz+08c].

Initialization Steps

The selection of an appropriate process model that defines the functional requirements for the parts of the later distributed system is the first step. Due to the fact, that such a choice can be a manual process, it should be supported by an experience factory providing knowledge and experiences – lesson learned – for the decision for or against a specific process model for the current need. The process model essentially bases on semantic metadata to allow the later automatic mapping of semantically described service functionalities to the functional requirements specified by the process model. With the chosen process model a set of concrete distributed systems is possible. In our measurement process characterization that means an essential measurement improvement.

$$A^{product \wedge resources}_{original} \cup A^{processs}_{original} = A^{product \wedge resources \wedge process}_{original}$$

After the experience-supported selection of an appropriate process model the second step of the presented approach is a selection of a quality model from a quality model repository. This is intended to be done automatically. For certain domains manual adaptations can be more efficient. A manual individualization of this predefined set of quality attributes as well as of their importance weighting is also possible. That means the experience basis in the measurement process establishes an *essential measurement improvement*

$$E^{criteria}_{threshold} \cup E^{formula}_{extension} \rightarrow E^{formula}_{repository} .$$

For these purposes an experience factory can be helpful again. As a result of this step *a process model and importance-ranked quality attributes are defined.*

Feasibility Check Steps

With this information process step three is able to determine whether enough available services exist to provide an acceptable amount of functionality demanded by the process model. If there is no acceptable coverage after the negotiation sub processes, then an abort probability based on already collected data can be computed. The user needs to decide whether he accepts the probability or not. If not the distributed system provision process will be aborted.

In the case of an acceptable coverage the runtime sub processes of step 4 can start.

The involved transformation of measurement results could be characterized as *remarkable measurement improvement* as

$$(Q^{ordinal\_scale}_{repository} \times E^{criteria}_{threshold}) \rightarrow (V^{ratio\_scale}_{data\_basis} \times U^{software\_unit}_{quasi\_standard})$$

The first of them determines the next process step to be executed following the process model. Therefore information about the last process steps can be taken into consideration to optimize the next process step execution. Exception handling in case of aborted pre-sub processes is a functional requirement and thereby should be covered by the process model itself.

Due to the fact that new services can be added to the service repository, another coverage check for the next process step is performed. Now, up-to-date service information, their evaluation values as well as the data of the quality model are available to identify the best service possible.

Selection Steps

The weighting of the quality attributes during the initialization delivered weighted attributes. This procedure is not intended to be performed during runtime, because the executed distributed system should not be interrupted (abort, costs …).

The result is a best possible distributed system based on the existing services as well as the specified quality model.

Operation and Evaluation Steps

Once the most optimal service is identified it can be executed and measured in parallel. These data are used to evaluate the last process step. The runtime sub processes are repeated until either all process steps of the process model are successfully executed or an abort due to missing services took place. Considering the quality assurance the modified kind of measurement tools can be described as *essential measurement improvement* as

$$T^{semi\_autmatic}_{one\_meas.\_phase} \rightarrow T^{autmatic}_{one\_meas.\_phase} \rightarrow T^{automatic}_{whole\_measurement}$$

including the derivation of the involved personnel that is used only for the first steps of infrastructure building as

$$P^{practitioner}_{measurement\_application\_staff} \rightarrow P_{initial}$$

The last step five of the presented approach covers the evaluation of the entire process being an input for the experience factory. It compares the achieved results with the desired ones.

5.3.2. Quality-Based Service Selection Core Process

In general the service selection has several steps. The first identifies all possible services according to the required functionality defined within the process model (during initialization phase). An additional step selects the identified quality model that specifies what quality aspects are useful for the intended usage and how important they are for the initiator of the application to be assembled. Manual adjustments are possible, but not necessary and are performed during initialization, too. Only in exceptional cases a manual adjustment during runtime is reasonable.

That means that the measurement itself using the QuaD^2 approach is changed as *essential measurement improvement* in the following manner

$$M_{measurement}^{assessment} \rightarrow M_{measurement}^{controlling}$$

Step three is the most important one and identifies the most appropriate service for the next process step to be performed during the selection process. It takes into account the weighted quality attributes as well the candidate service set whose elements fit the functional requirements of the current process step. Figure 98 shows a diagram presenting the underlying application flow of this special Service Selection Process.

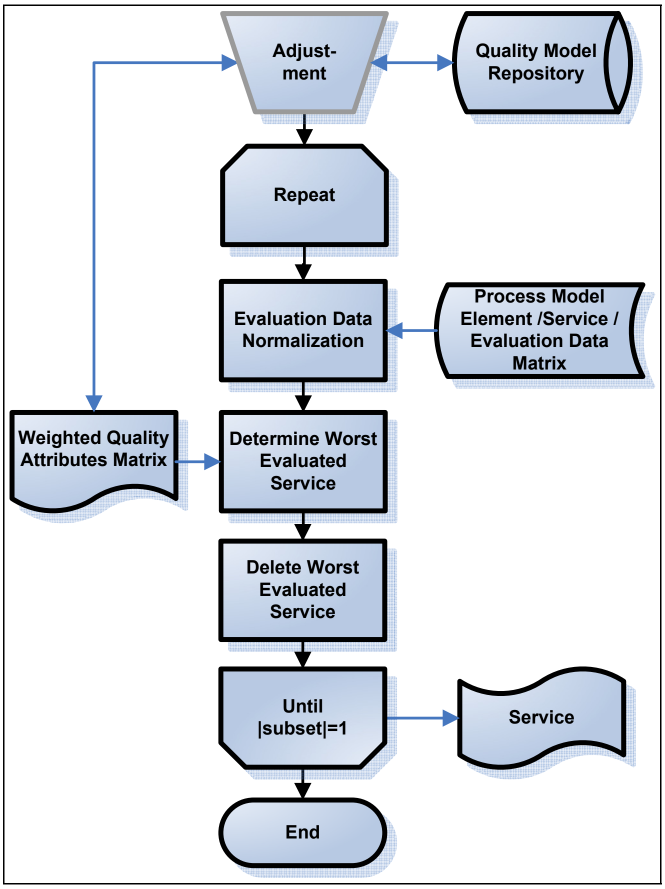

Adjust-ment
Quality Model Repository
Repeat
Evaluation Data Normalization
Process Model Element /Service / Evaluation Data Matrix
Weighted Quality Attributes Matrix
Determine Worst Evaluated Service
Delete Worst Evaluated Service
Until |subset|=1
Service
End

Figure 98: Service Selection Process

The weighted quality requirements matrix is manually created by selection needed quality attributes from a predefined set during initialization. The user has to weight the attributes in a normalized scale. For example he can decide to weight the cost of a service with 70% (0.7), the performance is considered to be less important (20% = 0.2) and size is weighted with 10% (0.1). All weights must sum up to 1.

Amongst others the calculation formula and normalization directive are stored for all quality attributes to be able to determine the qualitatively best service for the current need.

For the service selection process quite a few approaches have been taken into account. The general goal was to create a ranking based approach whereas the qualifiedly best service can be identified. A second arbitration was made to enable consideration of measures with different range of values: a normalization of service quality measures.

In doing so the normalization contains a transformation of values into a specific range [0 to 1]. On the one hand it enables comparison and combination of different measures on the other hand a shifting of values in the original set beyond the primarily values and thereby a changing of variance or deviation can influence the values of other services, even if their values haven’t changed at all. Normalization with the highest value among the candidates converts the value to the desired range [0 to 1] and the normalization keeps the ranking property. Due to the influence of the normalization range the values are only meaningful in comparison to identical normalized values which is sufficient for the desired ranking, of course.

To display the mentioned side effect Table 16 presents an example where three services are part of the selection process. A selected set of measures and specific weights for each measure constitutes the basis for the selection process. With the measured values for each service at a distinct point in time a normalization process is calculated and the aggregation of these values leads to the ranking of the three services.

In the course of time the quality measures for the several services can change, of course. Consequently, the presented example contains the second set of measured values. In that case the Response Time for a service has been worsening noticeable. In doing so the shifting of the range of values has the described side effect to the normalization process. The ranking in those measures stays consistent but the aggregation delivers a different result in comparison to the first dataset even though the values for service A and service B are still unchanged.

Table 16: Example of normalization implication

Point in Time X	Restorability (1 best)	Continued use of data (1 best)	Memory utilization	Response Time		
Service						
Service A	0.5	0.64	400 kB	30ms		
Service B	0.75	0.40	700 kB	20ms		
Service C	0.30	0.20	1100 kB	60ms		
Weight	0.20	0.20	0.20	0.40		
Normalized and weighted					Aggregation	Ranki
Service A	0.1	0.128	0.128	0.2	0.556	**2**
Service B	0.15	0.08	0.072	0.268	0.57	**1**
Service C	0.06	0.04	0.0	0.0	0.1	**3**
Point in Time Y						
Service A	0.5	0.64	400 kB	30ms		
Service B	0.75	0.40	700 kB	20ms		
Service C	0.30	0.20	1100 kB	200ms		
Normalized and weighted					Aggregation	Rank
Service A	0.1	0.128	0.128	0.34	0.696	**1**
Service B	0.15	0.08	0.072	0.36	0.662	**2**
Service C	0.06	0.04	0.0	0.0	0.1	**3**

The approach to overcame the described disadvantage has been identified in establish an elimination process. Since the worst service (regarding the measured values) is eliminated and the remaining values are re-normalized the described side effect is corrected.

Following the defined necessities and given data the service selection is formally described below. For the following formulas let PM be the chosen process model. Formula $f^{funct}(PM)$ specified in Formula 1 is used to determine the set of services E from the service repository. Each of them can deliver the functionalities specified within the chosen process model within formula 2.

$$f^{funct} : \text{Process model} \mapsto \{\text{Service}, ...\}$$ (Formula 1)

$$E = f^{funct}(PM)$$ (Formula 2)

Using the classic normalization approach presented in Formula 3, the evaluation values $v_{i,j}$ of quality requirements *j* defined in the quality model must be normalized for each service *i*. These $v_{i,j}$ are the measurement/simulation values to anticipate the optimal decision for the next process step.

$$v_i^{norm} = \frac{v_i - \min(v)}{\max(v) - \min(v)} * (\max_{norm} - \min_{norm}) + \min_{norm}$$ (Formula 3)

With the help of the weighted requirements matrix from the (maybe adjusted) quality model the last step – the identification of the optimal service according to the empirical data and the quality model – can be performed (see Formulas 4 to 8). Formula 4 adjusts the normalized evaluation values to ensure proper calculation. If *v=1* describes the best quality level then no adjustments are necessary, otherwise a minimum extremum is desired and *1-v* must be calculated.

$$f^{mm}(v) = \begin{cases} v & \text{,if a maximal } v \text{ is the best} \\ 1 - v & \text{,if a minimal } v \text{ is the best} \end{cases}$$ (Formula 4)

$$f^{eval}(e_i) = \sum_{j=0}^{n-1} f^{mm}\left(v_{i,j}^{norm}\right) \middle| e_i \in E \wedge n = |QM|$$ (Formula 5)

$$V = \left\{ f^{eval}\left(e_i\right) \middle| \forall e_i \in E \right\}$$ (Formula 6)

$$e^{worst} = e_{index} \mid index = \min\left(\{x \mid v_x = \min(V)\}\right) \wedge e_{index} \in E$$ (Formula 7)

$$E' = E \setminus e^{worst}$$ (Formula 8)

To determine the best evaluated service, Formulas 5 to 8 are repeated until E' contains only one element. It provides the needed functionality and is the most appropriate one according to the specified quality model.

After the service selection execution can occur and measurement about runtime behavior will be captured to get additional quality evaluations for this service.

Service Repository Management

As described in the general QuaD^2 Process the presented quality-driven service assembly framework requires a service repository and semantic descriptions and evaluation data about all available entities. This section introduces the Service Repository Managements Processes.

A first overview is given in the following Use-Case Diagram. Two different users are distinguished. The controller activates evaluation updates and the service provider who can either add or update a service. Every change of a service forces a new evaluation regarding all defined quality attributes.

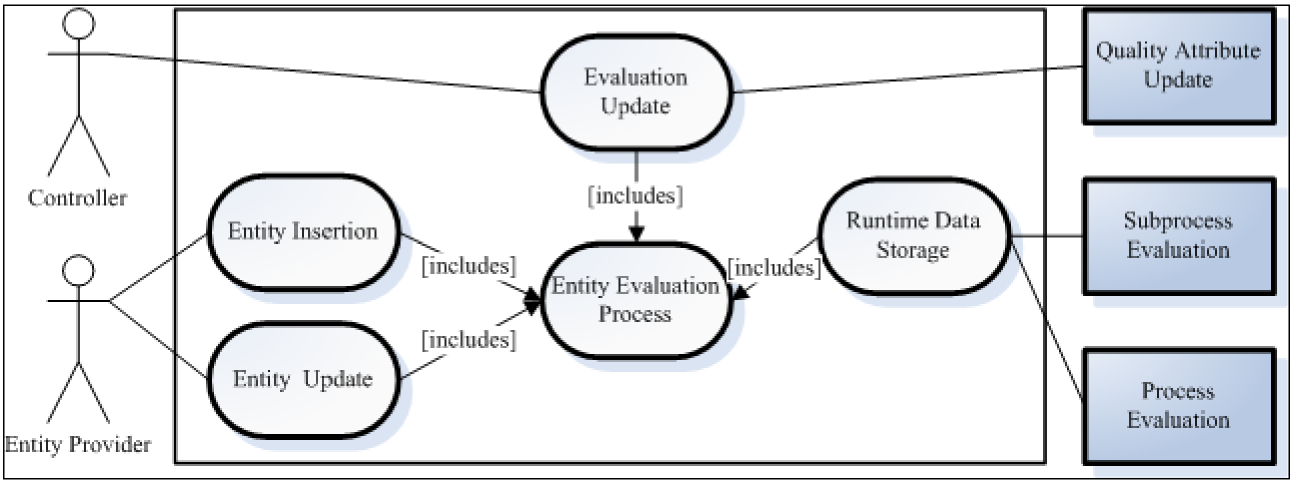


Figure 99: Use Case Diagram: Service Repository Management Use Cases

Amongst the described use cases, the service insertion is the major one. The insertion is divided into three steps – consistency check, standardization check and evaluation. The first and second focus on functionality and the third is about quality. As mentioned above, functionality-related issues are already discussed elsewhere. The quality-driven QuaD^2-Framework adds a new dimension and shifts the focus towards the Service Evaluation Process (see Figure 100).

The Service Evaluation Process uses the defined formulas for each quality attribute being stored in the Quality Attributes List to calculate the evaluation values for every service. Not for every attribute a mathematical formula is available, but at the attribute's definition time an evaluation procedure must be specified to allow quality assessment. Such evaluation procedures can be e.g. experiments, user surveys, or certain simulations.

Because initial evaluations can change over time, updates are necessary. For this purpose a runtime measurement is performed parallel to service execution. Event and time triggered service evaluations provide additionally empirical data. This continuous service evaluation ensures the throughout quality assurance during the QuaD^2-Processes and enables high quality software products [KernchenKunz$^+$07b].

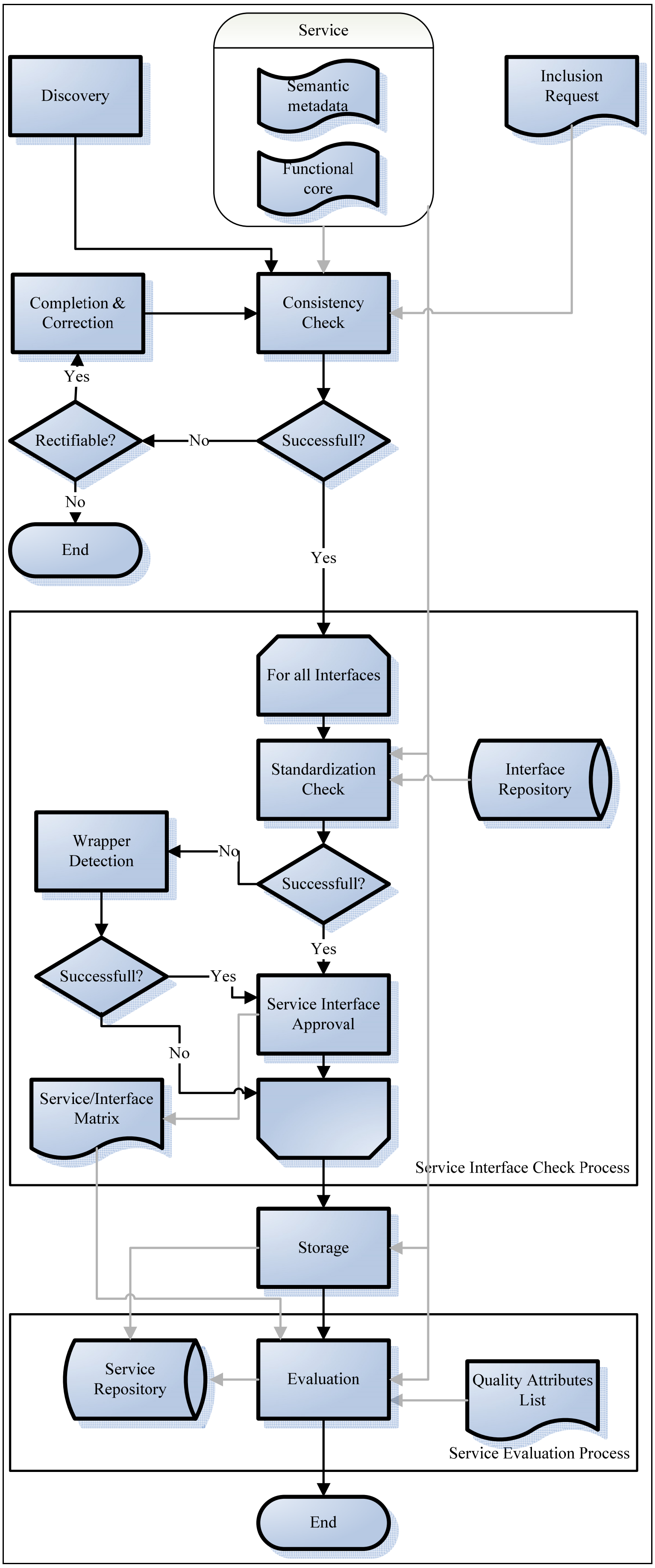


Figure 100: Service Evaluation Process

5.4. Graphical User Interface

According to the Use-case diagram of the whole measurement infrastructure (see Figure 101) different visualization opportunities has to be provided. Different views should be provided for developer, quality engineer, and project manager. Developer and product manager should be served with to analyze visualizations whereas the quality engineer view should enable to analyze the measured values in detail [Kunz+08d].

Since, the desired measurement framework takes into account these different roles of the software development process, different use cases could be identified and are considered for design of different graphical user interfaces.

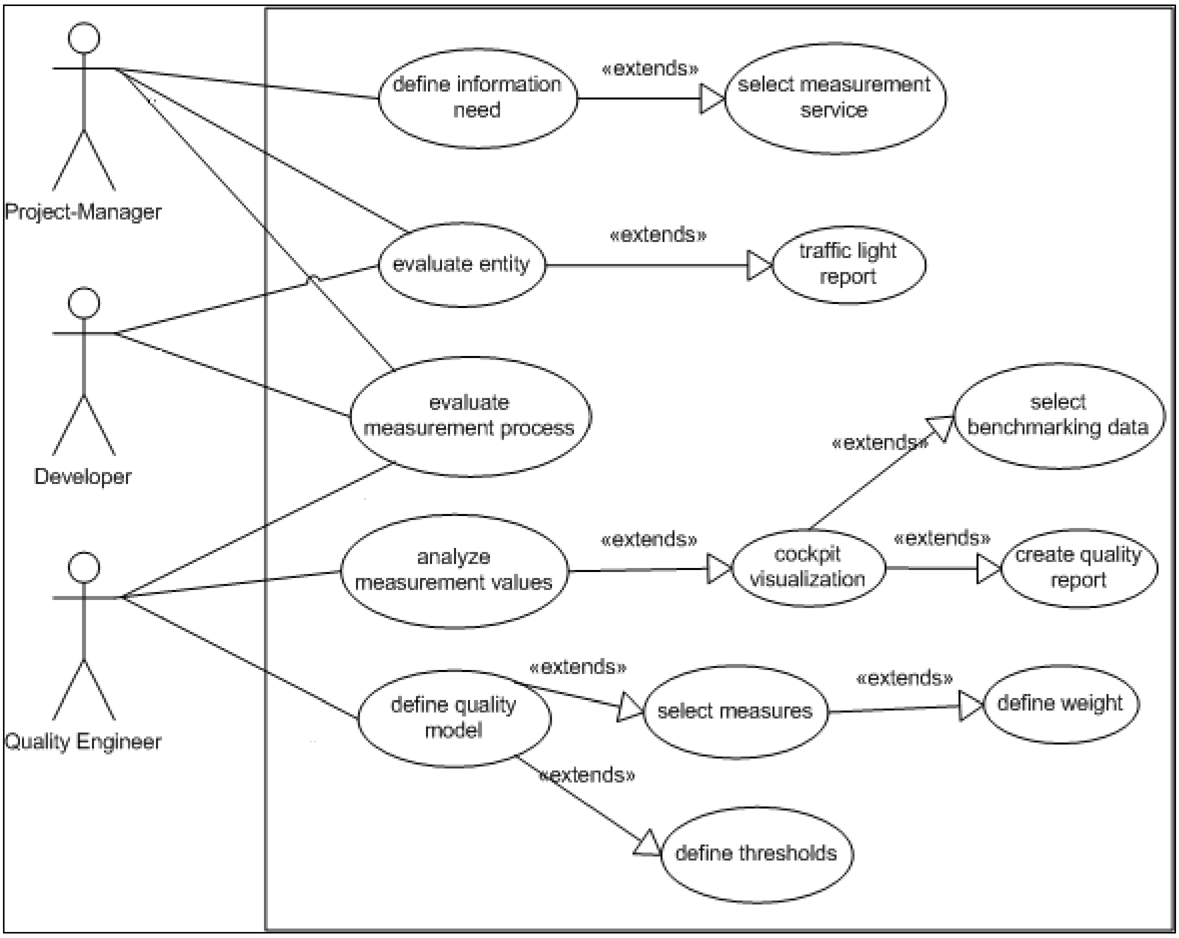


Figure 101: Use-case diagram of intended infrastructure

The selection and interplay of the different services are not the focus of the layer architecture. The figure is intended to present the approach of using plug-ins for development environments or project dashboards as presentation layer.

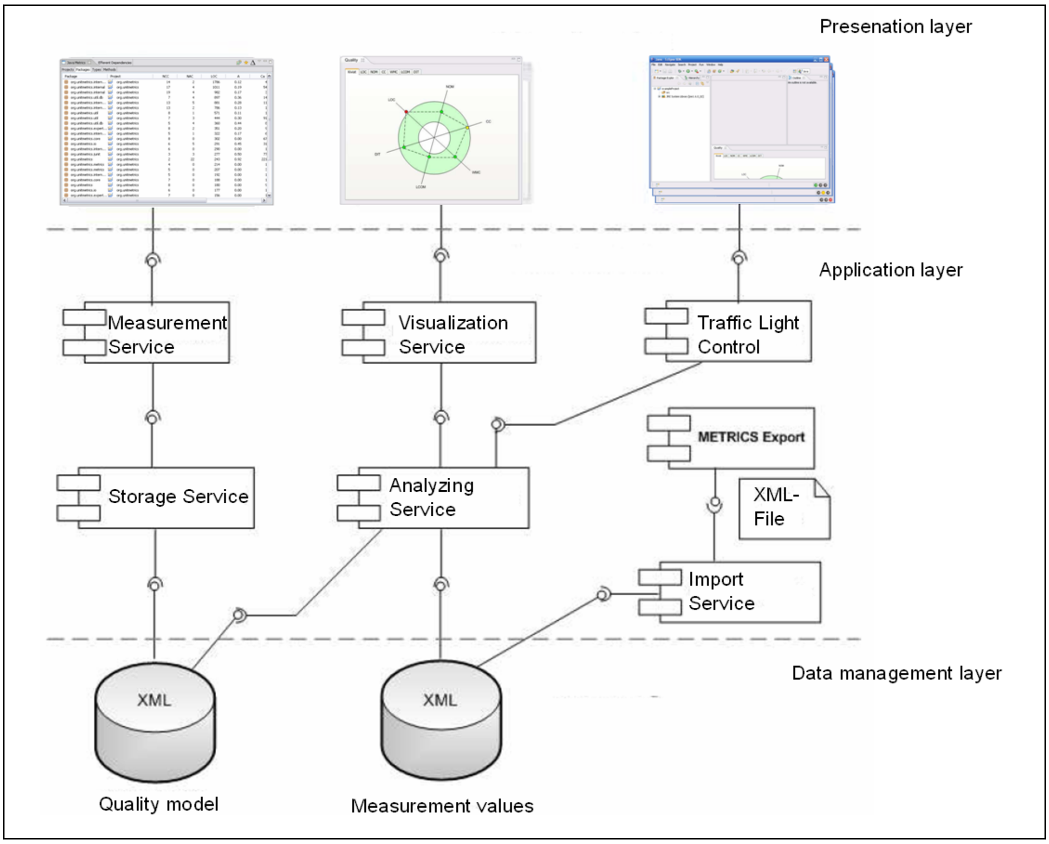


Figure 102: Architecture of intended framework

Providing analysis of measurement values is logically one of the most important parts of software measurement. The general goal is to give the different stakeholders an understanding of the measures quality attributes. Thereby it is an integrative part of every measurement activity but in literature one can identify manifold approaches of measurement data analysis [Kunz$^+$07a] [ZenkerKunz$^+$08b].

Since, the framework is intended to implement the ISO/IEC 15939 measurement standard for high diversity of use-cases the concept contains stakeholders with different view on measurement data.

To cover the different perspectives two visualization concepts have been implemented:

- Traffic Light Report
- Measurement Cockpit

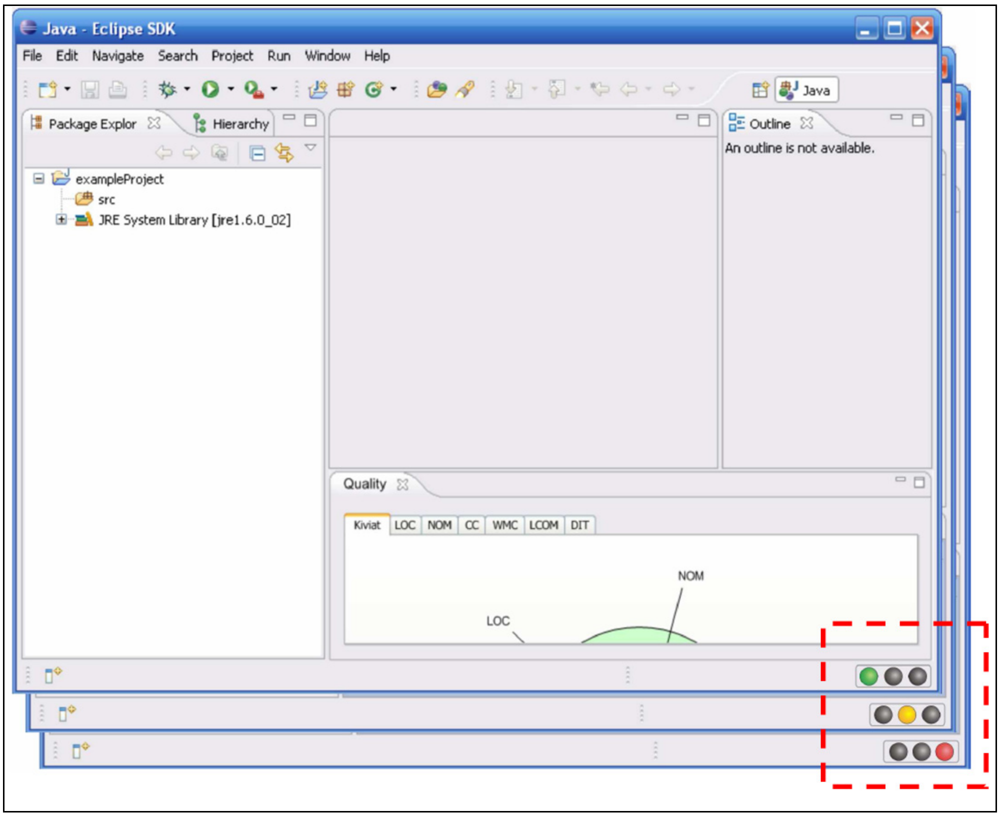
Taking into account the desired user of this type of measurement visualization it is very important to define in which way, from a CASE tool perspective, the different quality reports should be provided. For the integration of measurement reports in existing tools or new created infrastructures the providing as a service seems to be a consequential.

In consideration of the stakeholders it is obvious that in either case the traffic light report should be an integrative part of existing CASE tools.

Because of the quality engineers demand of a much more detailed view, complete analysis of all measured values, and the demand of creating own reports out of a huge amount of datasets, a stand-alone tool with a strong connection to the measurement database seems to be meaningful in this regard.

5.4.1. Traffic Light Visualization of Measurement Results

As the title of concept implies, the measured values should be mapped by using defined thresholds to present an easy to understand result. To derive this representation of quality out of the possibly high amount of measured data the methodology of the Goal-Question-Metric Method is used (see chapter 5.2.1. GQM Process Model).



An outline is not available.

Figure 103: Eclipse Plug-In Design [Hertel08]

In some cases, especially if the traffic light report indicates a lack of quality, the demand for a more detailed analysis constitutes the need for a visualization of the several measures with the same user interface.

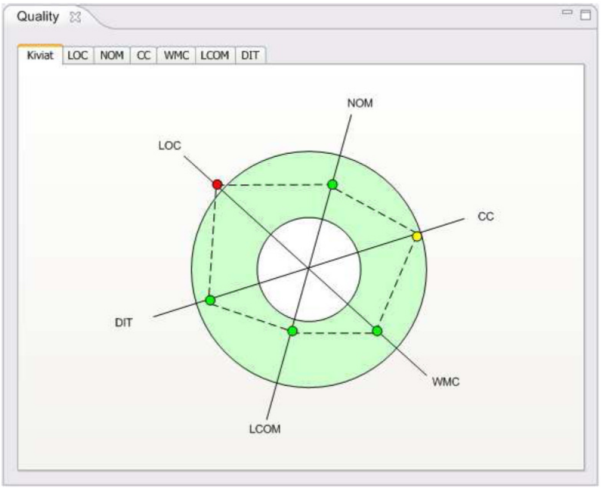


Figure 104: Kiviat diagram representation

A second aspect which is of interest in relation to traffic light report is the lapse of values. Once again of higher interest if the traffic light report switches to a non desirable state.

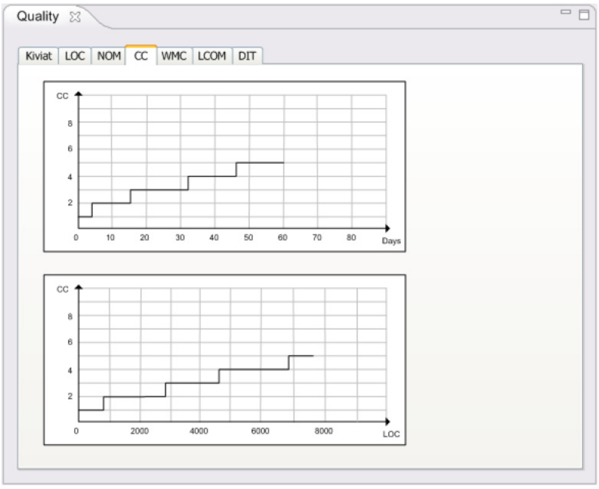


Figure 105: Measurement results over time

As mentioned above the traffic light visualization is desired to be integrated in CASE-tools for software development. This has been realized by implementing such plug-in for the Eclipse development environment [Eclipse07]. Eclipse is an open-source project aiming to provide an efficient platform for software development using various programming languages (e.g. Java, C++). The implementation of the plug-in has been done within a master-thesis project at the SML@b in 2007. A detailed description of the project and the outcome can be found in [Hertel08].

5.4.2. Cockpit for Measurement Results Analysis

Figure 106: Cockpit vision [Plenum06]

Not only in software measurement but more in more also in this area multi-screen cockpits gain higher importance for visualization purposes. As mentioned above especially for quality engineers the capabilities are helpful. Even if our own measurement service implementation so far only support object-oriented measures, the specification for a multi-screen measurement cockpit should take into account all available measures. Therefore the common determination of software measures into product, process, and resource seems to be adequate to define different views for the multi-screen cockpit. The main enhancement in comparison to Figure 102 is the enhancement of three view components in contrast to only one in the general plug-in concept.

One the one hand with arising technical solutions the limitation to one or two displays is outdated but on the other hand the limitation of the human mind to capture a limited amount of information at one time persists. That contradiction leads to the question how much information about a specific project or the process at one time is useful.

The general goal is not to provide a huge amount of information but to substantiate information where necessary.

Finally, the concept contains four views. Whereas one view provide merely communication and presentation instruments and the other three views provide measurement analysis and information. Namely the views are intended for:

- Project overview
- Measurement analysis
- Progression of measurement values over time / future trend / estimation
- Communication / presentation

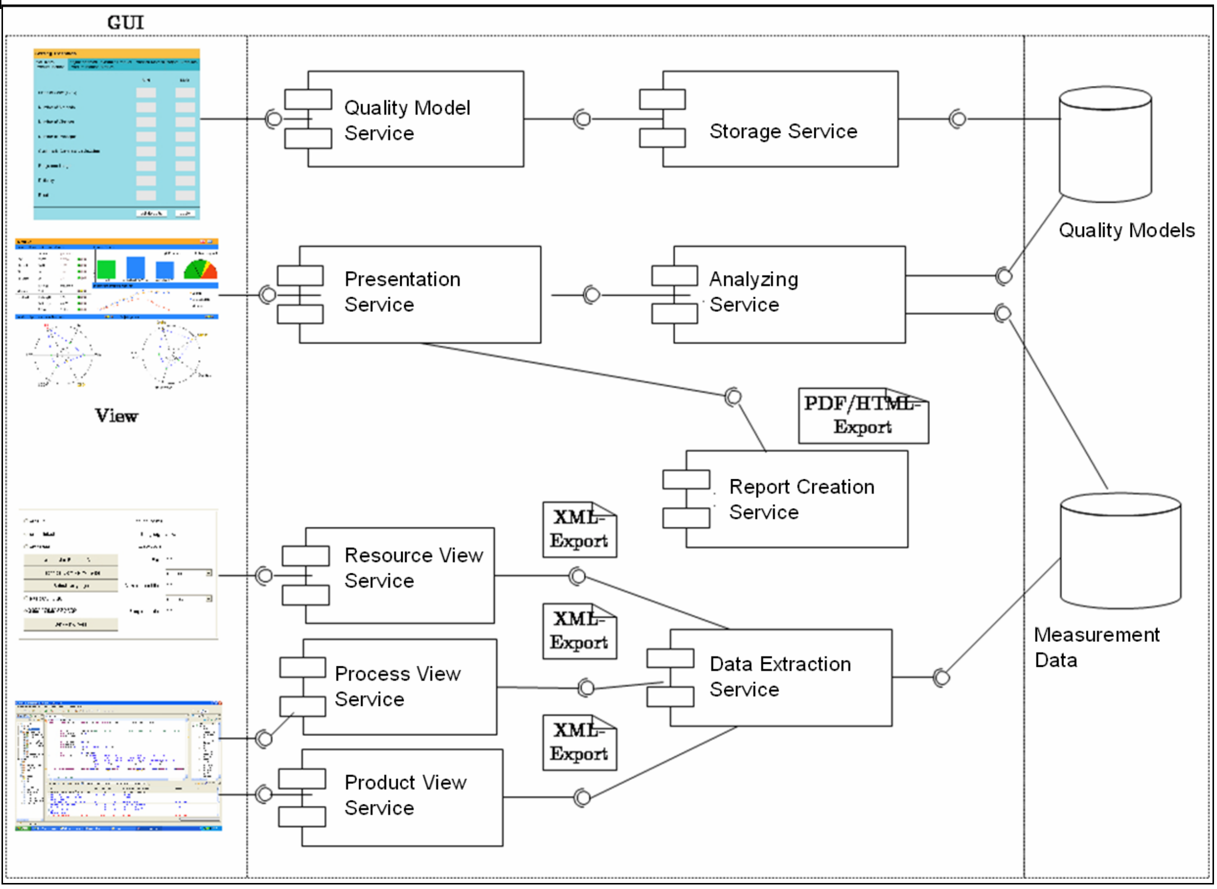


Figure 107: Architecture for cockpit view realization

By applying this approach a multi-screen measurement cockpit based on the presented building blocks and GUI elements could be implemented and the current setting is shown in Figure 108.

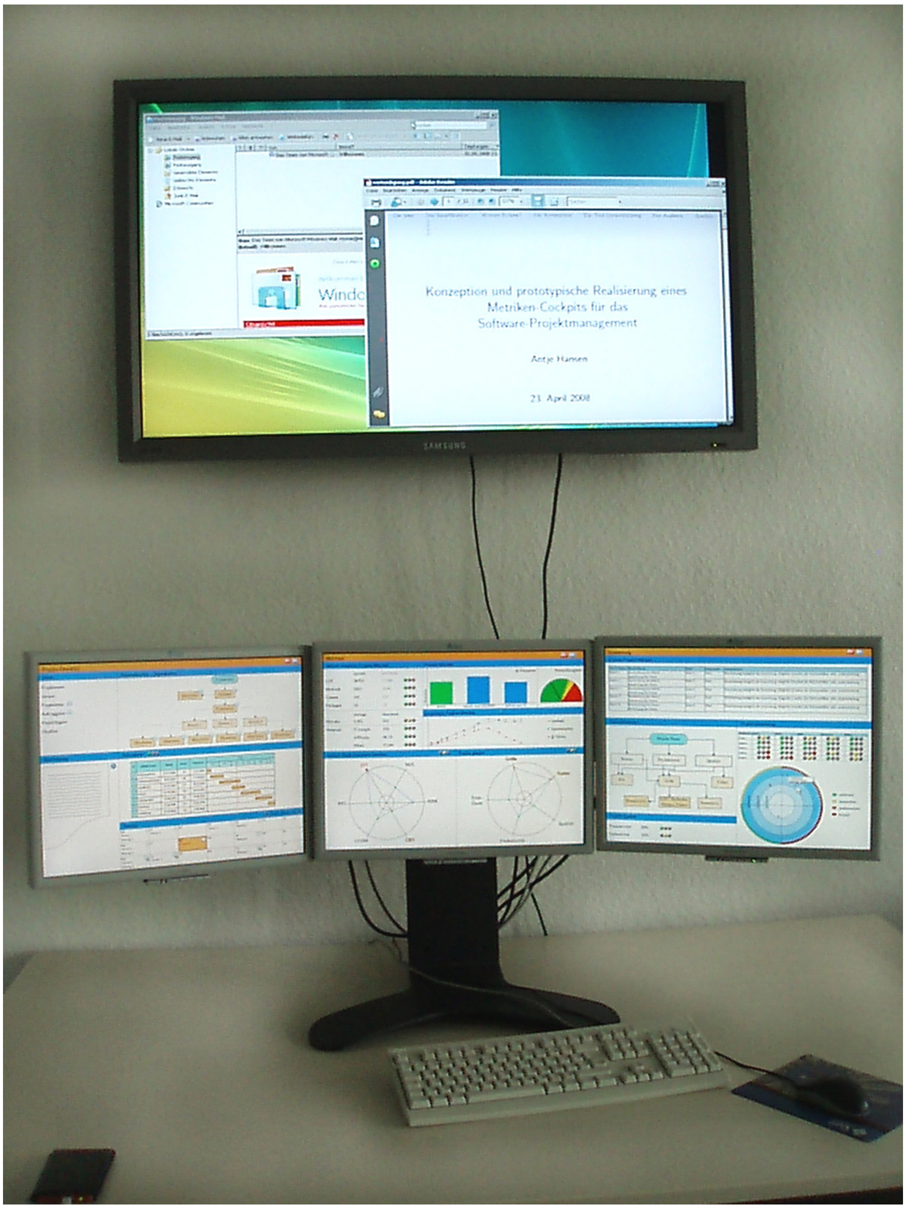

Konzeption und prototypische Realisierung eines
Metriken-Cockpits für das
Software-Projektmanagement
Antje Hansen
23. April 2008
SAMSUNG

Figure 108: Measurement cockpit at SML@b[Hansen08]

5.5. Service-oriented Measurement Database

The importance of a comprehensive storage of software measurement data is beyond controversy these days. Two major use cases can be identified in this regard:

- storage of measured data for analyzing
- provision of empirical or historical data for benchmarking or to support the analysis of new measured data

Based on the architecture in Figure 45 a service-oriented measurement database (SOMDB) has been created to build a central part of the desired service-oriented measurement infrastructure.

The data model of the database contains a key element: QualityModel. As described above a quality model contains a set of weighted attributes to provide an empirical base for analyzing measured values. Each created project holds an own set and the standard initialization values can be modified according to distinct project characteristics.

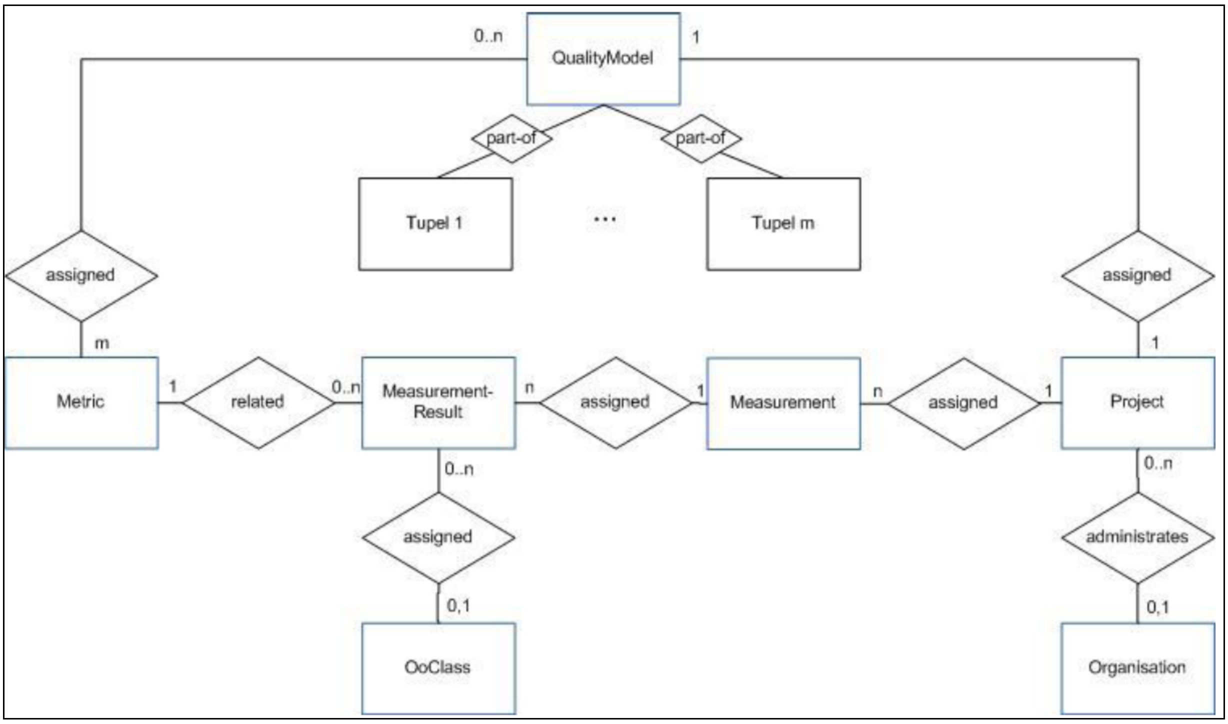


Figure 109: Entity-Relationship diagram of the SOMDB

For formal considerations the entity QualityModel (QMP) can be defined as:

$$\textbf{QMP} = (T_i \,|\, T_i = \{M_i, W_i, O_i, TG_i, TY_i\} \text{ with } M_i \text{ element } MP)$$

With MP as the set of all measured metrics in a distinct project the tuple QMP and:

M_i : Measure *M* element MP

W_i : Weight of M

O_i : optimum value of M

TG_i : target value

TY_i : threshold for acceptable values

The representation of this tuple in the Entity-Relationship diagram is shown in Figure 110, whereas for each defined project one tuple is existing.

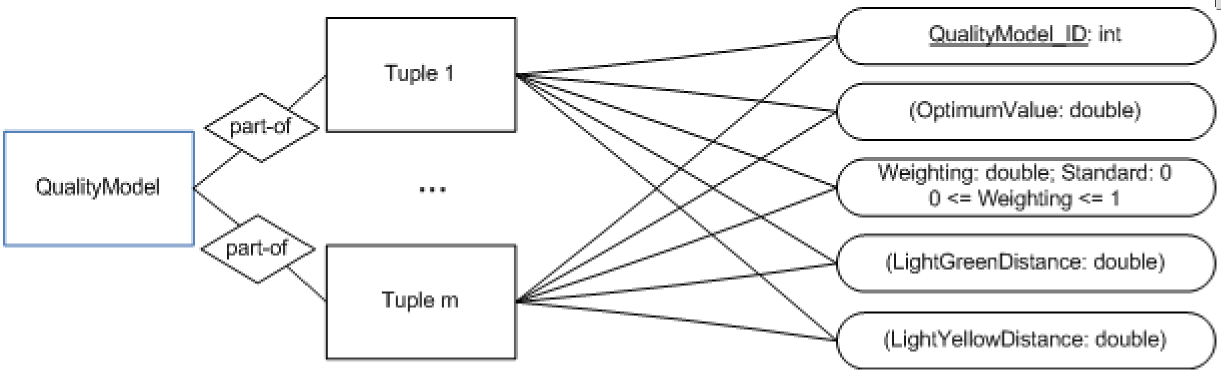


Figure 110: Detailed view on the QualityModel dataset

To provide a more graphically view on the threshold concept Figure 111 shows the three areas:

- ideal range: target value and a defined distance from that value (green color in traffic lights)
- accepted range: acceptable interval between the green interval and the second threshold (yellow color in traffic lights)
- beyond accepted range: unacceptable values (red color in traffic lights)

All three values (optimum, ideal threshold, accepted threshold) are defined as positive real numbers without any limitations.

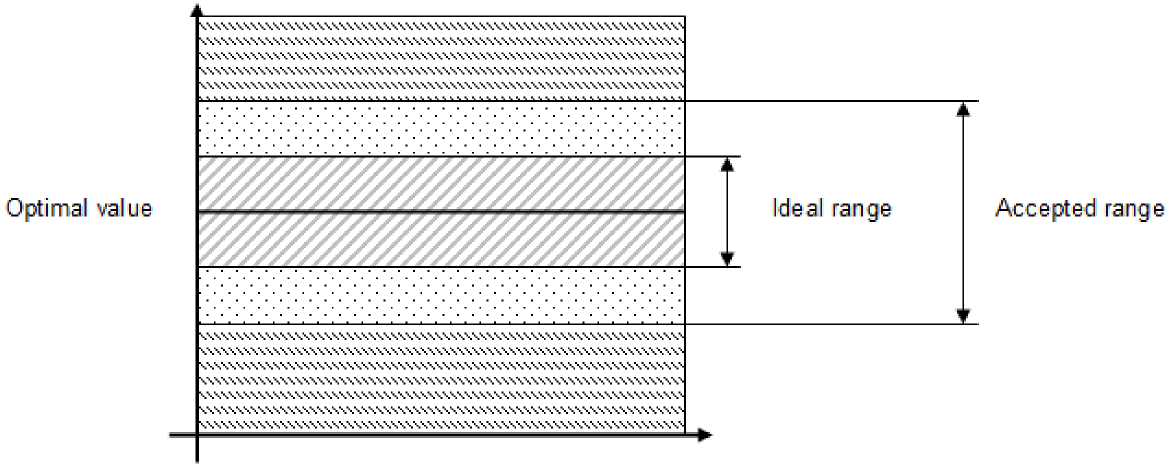


Figure 111: Threshold concept for measurement values

Another important part is the interface of the database and the thereby associated measurement data validation. To realize the validation data an approach by using XML Schema [HoriEuzenat+03] has been chosen. Thereby the structure of a valid document is defined and adjusted with the received one. In doing so the parameters of the interface are defined, too. The following figure shows in detail the structure of the created schema. Thereby, the statement of a project identifier and organization name is optional. All other attributes are required, namely project name, measurement tool name, and programming language. To this measurement dataset a various number of measurement results can be added. This measurement results are containing metric name, acronym, and the measurement value. Optionally, the identifier of the measured class can be added.

```
<?xmlversion = "1.0" encoding = "utf-8"?>
<xs:schema xmlns:xs = "http://www.w3.org/2001/XMLSchema">
   <xs:element name = "measurement">
      <xs:complexType>
         <xs:sequence>
            <xs:element ref = "measurementResult" maxOccurs = "unbounded"/>
         </xs:sequence>
         <xs:attribute name = "projectName" type = "xs:string" use = "required"/>
         <xs:attribute name = "projectID" type = "xs:integer" use = "optional"/>
         <xs:attribute name = "organisationName" type = "xs:string" use = "optional"/>
           <xs:attribute name = "measurementToolName" type = "xs:string" use = "required"/>
           <xs:attribute name = "programmingLanguage" type = "xs:string" use = "required"/>
      </xs:complexType>
   </xs:element>
<xs:element name = "measurementResult">
   <xs:complexType>
      <xs:sequence>
         <xs:element name = "metricName" type = "xs:string"/>
         <xs:element name = "metricAcronym" type = "xs:string"/>
         <xs:element name = "ooClass" type = "xs:string" minOccurs = "0"/>
         <xs:element name = "value" type = "xs:string"/>
      </xs:sequence>
   </xs:complexType>
</xs:element>
</xs:schema>
```

Figure 112: XML-Schema definition

In Figure 113 an example of a measurement file with two measurement results is presented.

At each time the web service is called the measurement file is transferred and the data is stored. In doing so, a latching or a buffer in the web client is not necessary and the size of the file only depends on the number of measurement results to be stored.

```
<?xml version = "1.0" encoding = "ISO-8859-1" ?>
<measurement projectName = "SMLab project" projectID = "1" organisationName = "SMLab"
measurementToolName = "OOMJ" programmingLanguage = "Java">
          <measurementResult>
                    <metricName>Number Of Children</metricName>
                    <metricAcronym>NOC</metricAcronym>
                    <ooClass>checktransaktion</ooClass>
                    <value>4</value>
          </measurementResult>
          <measurementResult>
                    <metricName>Lines Of Code</metricName>
                    <metricAcronym>LOC</metricAcronym>
                    <value>121324</value>
          </measurementResult>
</measurement>
```

Figure 113: Example of a measurement file

5.5.1. Measurement Data Analysis

As a result of the survey about existing measurement spreadsheets and databases (see chapter 3.2. Software Measurement) the visualization of measurement data and the thereby provided analysis possibilities are a key success factor for software measurement in general. This defines the goal of drawing meaningful conclusions out of the created measurement database.

The major concept in this regard is the visualization by using kiviat diagrams (as presented in Figure 35). Bearing the capability of presenting different measures in one chart, alongside with a defined quality model or other measurement data makes this approach the agent of choice.

In Figure 114 the blue line visualizes the measured value for a distinct project. The desired values for each measure are drawn with the green line. The traffic light behind each measure represents the analysis regarding the defined thresholds for this specific project. This has been done sine drawing all three lines of the quality model makes the emerging figure very confusing.

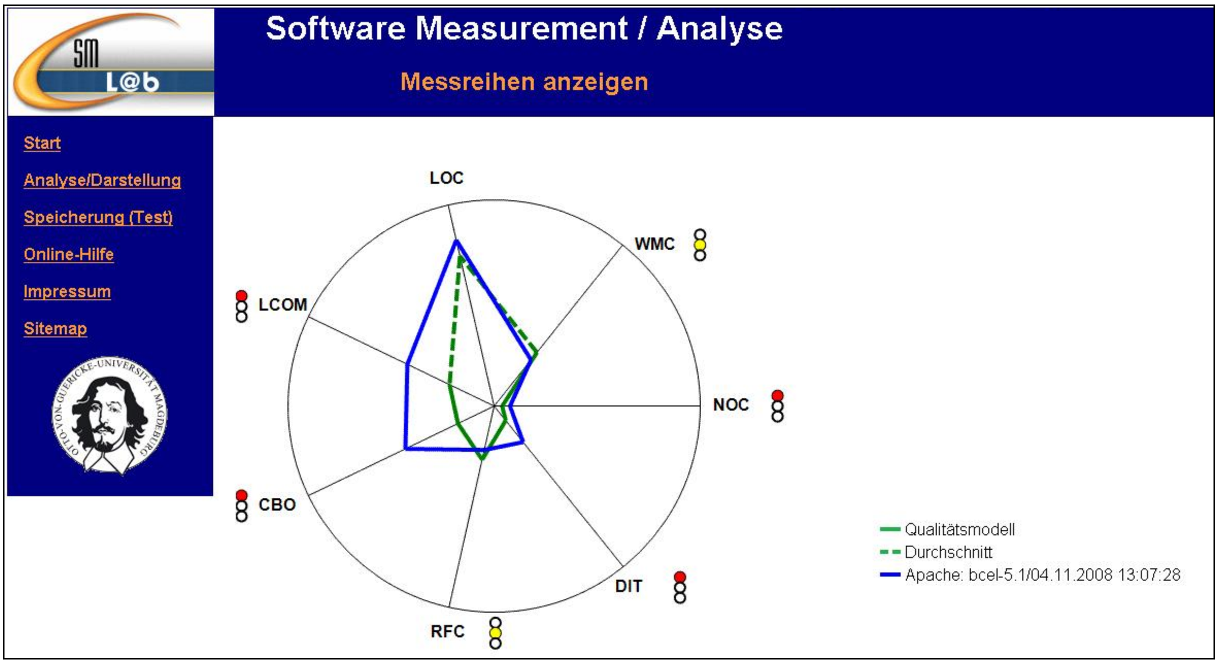
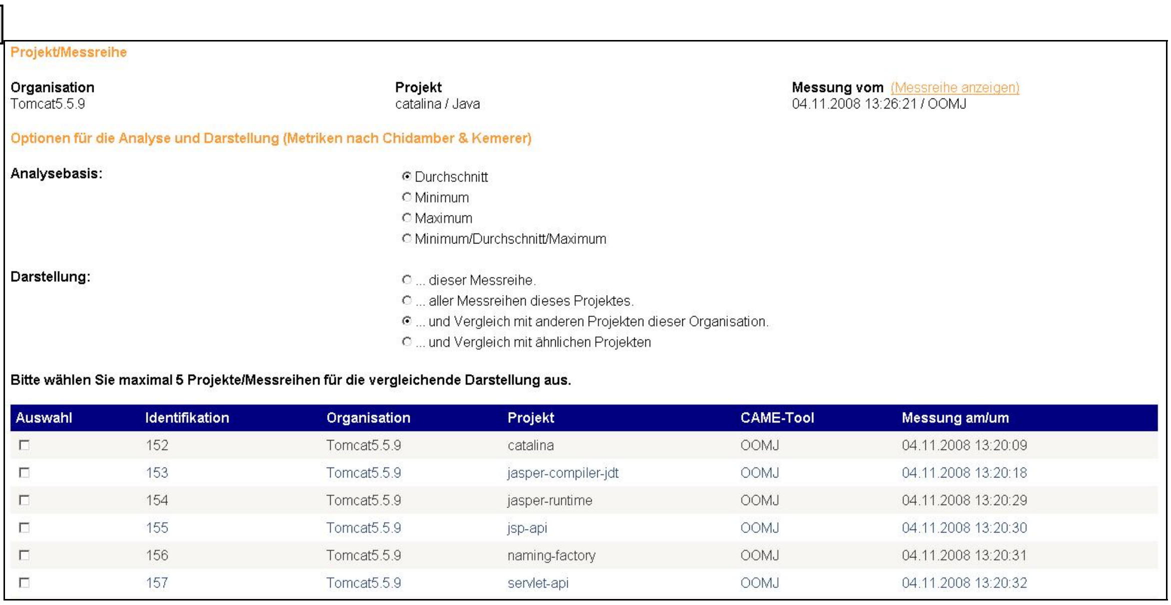


Figure 114: Overview about the Service-oriented Measurement Database

Since, the quality model is a generic one sometimes the desired values are out of reach or beside a reachable level. In this case a comparison with other real projects provides a more realistic view on software quality. For this purpose the SOMDB contains the possibility of choosing similar projects as benchmark on the basis of included measures. Logically, one has to select a set of measures which are used as the basis to calculate similarities and this set of measures should not included the measure to analyze. Otherwise one can only identify if other project reaches a specific characteristic with less Lines of Code or a lower Depth of Inheritance Tree but assertions about the target measure is not possible.



Projekt/Messreihe

Organisation
Tomcat5.5.9

Projekt
catalina / Java

Messung vom (Messreihe anzeigen)
04.11.2008 13:26:21 / OOMJ

Optionen für die Analyse und Darstellung (Metriken nach Chidamber & Kemerer)

Analysebasis:
● Durchschnitt
○ Minimum
○ Maximum
○ Minimum/Durchschnitt/Maximum

Darstellung:
○ ... dieser Messreihe.
○ ... aller Messreihen dieses Projektes.
● ... und Vergleich mit anderen Projekten dieser Organisation.
○ ... und Vergleich mit ähnlichen Projekten

Bitte wählen Sie maximal 5 Projekte/Messreihen für die vergleichende Darstellung aus.

Auswahl	Identifikation	Organisation	Projekt	CAME-Tool	Messung am/um
□	152	Tomcat5.5.9	catalina	OOMJ	04.11.2008 13:20:09
□	153	Tomcat5.5.9	jasper-compiler-jdt	OOMJ	04.11.2008 13:20:18
□	154	Tomcat5.5.9	jasper-runtime	OOMJ	04.11.2008 13:20:29
□	155	Tomcat5.5.9	jsp-api	OOMJ	04.11.2008 13:20:30
□	156	Tomcat5.5.9	naming-factory	OOMJ	04.11.2008 13:20:31
□	157	Tomcat5.5.9	servlet-api	OOMJ	04.11.2008 13:20:32

Figure 115: Analysis by using similar projects regarding a specific measure

The values of the selected similar projects are drawn together with the main project in a kiviat diagram, too.

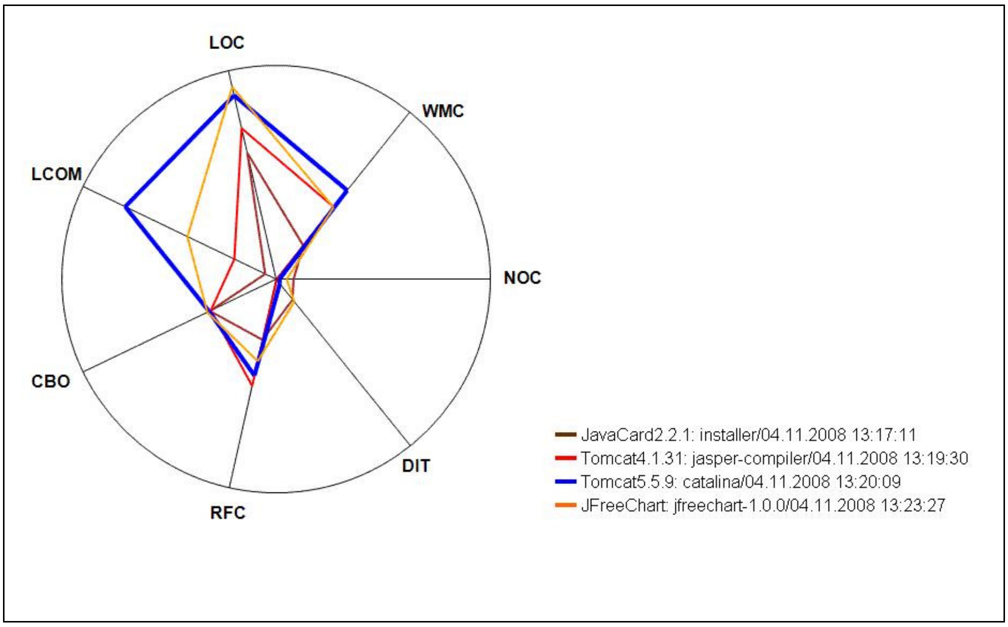


Figure 116: Kiviat Diagram with selected projects

At the moment the database contains 15.000 measures java classes, 150.000 measured java methods, and over 400 million measured lines of code. With this huge amount of comparative data the SOMDB is one of the biggest in her sort and a very powerful tool for benchmarking approaches.

5.6. Mapping to Measurement Paradigms

The kernel idea constructing proactive measurement infrastructures is based on the so-called QuaD^2 framework. This framework can be implemented using various technologies as e.g. ontology's, web services and agents. The presented quality-driven approach uses semantic descriptions for processes automation and supports different quality models and quality attribute evaluations. The easy extensibility of process models, services, interfaces and quality models makes the presented framework deployable for many fields of application.

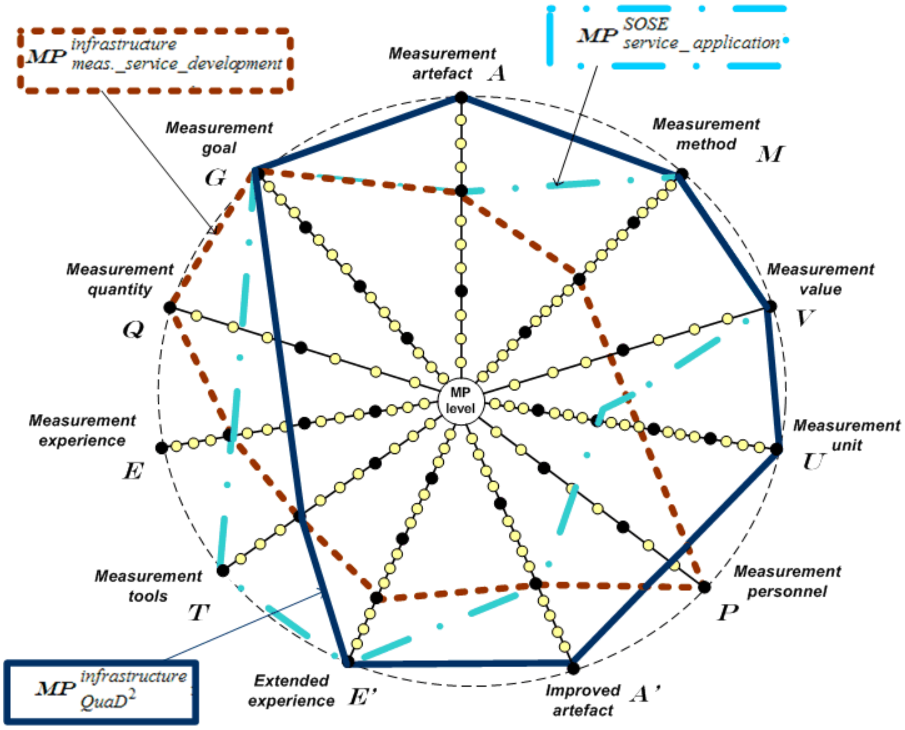


Figure 117: Software measurement process levels including the QuaD² approach

6. Summary and Future Work

The final chapter of the thesis recapitulates this work, provides a summary and a future perspective of the presented topic.

The thesis in hand presents the undertaken steps of a PhD project to answer the main target:

How should an infrastructure for software measurement be arranged, to implement the software measurement process of an organization?

Chapter 1 draws the point of origin of the thesis. The increasing importance of software engineering in general and software measurement in detail constitutes a demand for a research work in this area.

The foundations of software measurement and existing approaches are presented in chapter 2. Thereby an overview about general aspects of software measurement is presented. In doing so the elements of a software measurement systems and different paradigms for software measurement are presented.

The implementation of these approaches as a measurement program is described in chapter 3. Especially the drawbacks of existing solutions and the resulting success rate are discussed and the reasons are analyzed.

Additional, existing approaches to tackle the described drawbacks and an evaluation of existing measurement paradigms guides the way to a service-oriented measurement infrastructure.

For the provision of a new type of measurement approach two different drivers were identified:

a) customer asks for tailored functionality to combine components of different tool manufactures and to avoid high acquisition costs
b) Measurement tool producer’s needs to gain access to new sales markets which is intended to be reached by provide services combined with the possibility of new license models e.g. pay per use.

To introduce this topic, chapter 4 contains a presentation of service-oriented IT architectures, in a nutshell. An important part of this chapter is the analysis of SOA-capability of existing measurement tools. This analysis is realized by taking into account different viewpoints. A detailed survey of tool manufactures about the capabilities of existing tools and future strategy is validated by an independent analysis about SOA capabilities of existing measurement tools. This analysis provides valuable input for the framework for a service-oriented measurement infrastructure.

The review of the state-of the art, the need and relevance stated by a survey, the analysis of SOA, and the determination of the SOA capability of existing measurement tools leads to the framework described in chapter 5.

The implementation of such infrastructure based on object-oriented metrics constitutes a proof of concept with a highly distributed type of measures in practical software measurement.

The single elements are presented as well as prototypical implementations of measurement components which are able implement all desired elements of the service-oriented measurement infrastructure. The inclusion of graphical user interfaces for a very broad range of users (from project management to quality engineers) reason a high distribution into a software engineering organization in comparison with existing solutions focused on single functionality.

Another major component, the service-oriented measurement database, is described in detail in chapter 5.

Some of the presented components provide a benefit away front the presented use case of a Service-oriented measurement infrastructure. Especially the component for a quality driven assembly of services QuaD² creates benefit for all service-oriented solution away from the presented use-case.

Subsuming some elements missing in available solutions have been achieved by own investigations:

- Creation of an ontology for object-oriented metrics including a web service for standardized machine readable interface
- BPMN based Implementation of the ISO/IEC 15939 measurement process standard for a throughout realization of measurement processes
- Application of the OASIS Reference Model for Service-Oriented Architecture 2.0
- Integration architecture for existing solutions and an integration of the infrastructure into an IDE by using plug-ins
- the mentioned QuaD² framework for quality driven service assembly

The Validation of the presented approach has been done by different approaches. Surveys and case studies regarding existing solution prove their relevance for establishing new solutions in this regard.

Prototypical implementation and validation by publishing them in various papers and thereby applying Delphi studies [RoweWright01] on them implies a validate approach of the framework.

Applying the Delphi approach to the complete framework or a validation approach by implementing the whole architecture beyond a prototypical solution for a practical validation and empirical studies about the usage would go beyond the scope of the thesis in hand and has to be kept for future work in this research area.

To reach the described goals the thesis mainly deals with the following areas:

- the challenges and the potential of provide measurement services instead of monolithic measurement tools
- measurement infrastructure alongside a defined measurement process
- semantic description of service to provide automated assembly
- design of service-oriented architectures on the basis of quality indicators
- process evaluation as input for infrastructure design

Future Perspective

Based on the evaluation of the technical solution two main application scenarios are reasonable:

- Establishing of public service provider to provide single services as broker and which will be paid by different license models (e. g. pay per use, pay per month, flat rate). Commercial services can be provided alongside with open-source services. The functionality of each service is described in a semantic manner and the service provider measures the services in use, in this case the service selection can be done by functional and non functional characteristics
- The main technology is adopted by large development organizations which build their own infrastructures inside their intranet. According to needed measurement functionality services from measurement tool producers are bought and integrated into the internal infrastructure.

Both use cases provide promising perspectives for the future of software measurement tools. The first one is able to enable small and mid size companies to implement software measurement because high acquisition costs and thereby unclear cost/benefit ratio is solved.

The second case enables a throughout solution regarding software measurement in large scale development organizations and provide one solution for a hole measurement process, consolidates the knowledge but providing different graphical user interfaces, analysis possibilities and different level of detail for different stakeholders.

In both cases two facts can be assured from whereas benefits for the framework are expected:

- More functionality is provided as web services owing semantic descriptions for an automated assembly. That would enhance the applied measurement field to other than object-oriented measures
- more empirical data about measurement processes, web services and measurement results can be collected and applied for an enhancement of measurement data analysis and web service evaluation

In both cases the application to other measurement areas and other application areas of service-oriented architectures can be identified as beneficial.

Beside the enhancement with more functionality and empirical data a framework like the presented one needs acceptance beside the scientific world. But economic success not always relates to technical capabilities or applied standards. More often success in commercial business relates to support and publicity by perceived communities.

To promote the framework he has been published at various conferences and workshops. Additionally, the framework is supported among other by the German speaking user association for software metrics and effort estimation and by the central European computer measurement group at their workshop for measurement aspects of service-oriented architectures.

Bibliography

[AbrLaf+99] 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.

[AbrKunz+03] Alain Abran, Martin Kunz, Reiner R. Dumke, and René Braungarten. The prototypical web-based implementation of the QEST model. Proc. of the IWSM2003, Montréal, Canada, 23.-25. September, 2003, pp. 82-92, ISBN 3-8322-1880-7

[AbranMoore+04] 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.

[AbranOligny+00] Alain Abran, S. Oligny, Charles Symons: COSMIC FFP and the World Wide Field Trials Strategy. In R.R. Dumke and A. Abran editors: New Approaches in Software Measurement. Springer Publishing. Berlin. 2000

[AlvesAssaf+07] Alves, Alexandre ; Arkin, Assaf ; Askary, Sid ; Barreto, Charlton ; Bloch, Ben ; Curbera, Francisco ; Ford, Mark ; Goland, Yaron ; Guízar, Alejandro ; Kartha, Neelakantan ; Liu, Canyang K. ; Khalaf, Rania ; König, Dieter ; Marin, Mike ; Mehta, Vinkesh ; Thatte, Satish ; Rijn, Danny van d. ; Yendluri, Prasad ; Yiu, Alex: Web Services Business Process Execution Language Version 2.0. OASIS standard. http://docs.oasis-open.org/wsbpel/ 2.0/OS/wsbpel-v2.0-OS.html. Version: April 2007

[AngMarOls+03] Maria de Angeles Martin; Luis Olsina. Towards an Ontology for Software Metrics and Indicators as the Foundation for a Cataloging Web System/ GIDIS, Department of Informatics, Engineering School at UNLPam, LaPampa. Argentina. 2003.

[ASG08] Allen Systems Group, Inc.: ASG-Rochade – Product Details. http://www.asg.com/products/product_details.asp?code=ROC&id=50, Naples (USA), cited December 2008.

[AuerGraser+03] M. Auer, B. Graser, and S. Biffl, "A survey on the fitness of commercial software metric tools for service in heterogeneous environments:

Common pitfalls," in Proceedings of the Ninth International Software Metrics Symposium (METRICS'03), 2003, p. 144.

[BakerHantos03] Emanuel R. Baker and Peter Hantos. Implementing a successful metricsp rogram: 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.

[Balzert08] Helmut Balzert. Lehrbuch der Softwaretechnik.-.Softwaremanagement. Spektrum Akademischer Verlag. Heidelberg. 2008.

[BarrosDumas+05] A. Barros. M. Dumas, P. Oaks. A Critical Overview of the Web Service Choreography Description Language. BPTrends. July 2005

[Basili85] Victor Basili. Quantitative Evaluation of Software Engineering Methodology. Proc. Of the First Pan Pacific Computer Conference, Melbourne (Australia), 1985.

[BasCal+92] Basili, V.; Caldiera, G.; McGarry, F.; Pajerski, R.; Page, G.; Waligora, S.: The Software Engineering Laboratory - An Operational Software Experience Factory. In: Proc. of the 14th International Conference on Software Engineering, ACM, pp. 370-381, Melbourne (Australia), 1992.

[BasCal+94] Basili, V.; Caldiera, G; Rombach, D.: Experience Factory. In: Encyclopedia of Software Engineering. Editor: Marciniak, J. J., Volume I, pp. 469 - 476, John Wiley & Sons, New York (USA), 1994.

[BasiliWeiss84] Victor R. Basili and D.M. Weiss. A methodology for collecting validsoftware engineering data. IEEE Transactions on Software Engineering,10(6):728–738, June 1984.

[BasiliSelby+86] Victor R. Basili, Richard W. Selby, and David H. Hutchens. Experimentationin software engineering. IEEE Transactions on Software Engineering,SE-12(7):733–743, July 1986.

[Basili96] 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.

[BassClements+03] L. Bass, P. Clements, R. Kazman. Software Architecture in Practice. Addison-Wesley Professional 2nd edition. 2003.

[BauerMüller04] Bauer, B., Müller, J.: Methodologies and Modeling Languages. In: Agent-Based Software Development, Luck, M.; Ashri, R. and d'Inverno, M. (Editors), Artech House, Boston, pp. 77-131, 2004.

[Bergin93] Bergin, T. J.: Computer-Aided Software Engineering: Issues and Trends for the 1990s and Beyond. Idea Group Publishing, Harrisburg (USA), 1993.

[Bernstein97] Bernstein, P. A.: Repositories and Object-Oriented Databases. Datenbanksysteme in Büro, Technik und Wissenschaft, In: Dittrich, K.R. und Geppert, A. (Editors.): Proceedings of BTW Conference 1997 Ulm, p. 34-46, Springer-Publishing, Berlin (Germany), 1997.

[BerHar$^+$97] Bernstein, P.; Harry, B.; Sanders, P.; Shutt, D.; Zander, J.: Microsoft Repository. In: Proceedings of the 23rd VLDB Conference, Athens (Greece), pp. 3-12, 1997.

[Berri90] Beeri, C.: A formal Approach to object-oriented Databases. Data and Knowledge Engineering, Volume 5, Nmbr. 4, pp. 353-382, 1990.

[BerryJeffery00] Michael Berry and Ross Jeffery. An instrument for assessing softwaremeasurement programs. *Empirical Software Engineering*, 5(3):183–200,November 2000.

[BoehmBrown$^+$76] Barry Boehm, J.R. Brown, M. Lipow. Quantitative Evaluation of Software Quality. Proceedings of the International Conference on Software Engineering. Pg. 592-605. San Francisco. CA. USA. 1976

[BoehmAbts$^+$00] Barry Boehm, C. Abts., et al. Software Cost Estimation with Cocomo II. Prentice Hall PTR. Upper Saddle River. NJ. USA. 2000

[Boehm06] Barry W. Boehm. A view of 20th and 21st century software engineering.In ICSE '06: Proceeding of the 28th international conference on Softwareengineering, pages 12–29, New York, NY, USA, 2006. ACM Press.

[BoothHugo$^+$04] Booth, David ; Haas, Hugo ; McCabe, Francis ; Newcomer,Eric ; Champion, Michael ; Ferris, Chris ; Orchard, David: Web Services Architecture. W3C Working Group Note. http://www.w3.org/TR/ws-arch/. February 2004

[BourqueOligny$^+$07] Bourque, P.; Oligny, S.; Abran, A.; Fournier, B.: Developing Project Duration Models in Software Engineering. Journal of Computer Science and Technology, 22(3). 2007.

[Bradley98] Bradley, N.: The XML Companion. Addison-Wesley, New York (USA), 1998.

[BraFarKunz$^+$06] Braungarten, R.; Farooq, A.; Kunz, M.; Schmietendorf, A.; Dumke, R.: Applying Service-Oriented Software Measurement to Derive Quality Indicators of Open Source Components. UPGRADE - The European

Journal for the Informatics Professional, Vol. VII, No. 1, February 2006, ISSN 1684-5285

[BraKunz^{+}05a] Braungarten, R.; Kunz, M.; Farooq, A.; Wille, C.; Schmietendorf, A.; Dumke, R.: A Metrics Data Base Maturity Model. Proceedings of the 9th IEEE International Multi Topic Conference (INMIC2005), National University of Computer and Emerging Sciences, Karachi/Pakistan, December 2005, ISBN 0-7803-9430-5

[BraKunz^{+}05b] Braungarten, R.; Kunz, M.; Farooq, A.; Dumke, R.R.: Towards Meaningful Metrics Data Bases. Proc. of the IWSM05, September 12-14, 2005, Montreal, Shaker Publishing. Aachen. 2005

[BraKunz^{+}05c] René Braungarten, Martin Kunz, and Reiner R. Dumke. An Approach to Classify Software Measurement Storage Facilities. Preprint Nr. 2, Computer Science Departement, University Magdeburg, 2005

[Braungarten07] René Braungarten . A The SMPI Model: A stepwise process model to facilitate software measurement process improvement along the measurement paradigms. PhD thesis, Otto-von-Guericke University of Magdeburg, 2007.

[BuglioneAbran99] Luigi Buglione, Alain Abran. Multidimensional Software Quality Measurement Models: A Tetrahedron-based Design. In: Dumke R, Abran A, editors. Software Measurement: current trends in research and practice. Deutscher Universitäts- Verlag GmbH, 1999.p.93-107.

[BunDek08] Manfred Bundschuh, Carol Dekkers. The IT Measurement Compendium: Estimating and Benchmarking Success with Functional Size Measurement. Springer Publishing. Berlin. 2008

[ChaBer08] Chappell, David ; Berry, David: Next-Generation Grid-Enabled SOA: Not Your MOM's Bus. In: SOA Magazine XIV, January. 2008

[Chappell06] Chappell, David: SOA and the Reality of Reuse http://www.davidchappell.com/HTML_email/Opinari_No16_8_06.html, Cited December 2008

[ChiKem94] 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.

[Chen76] Chen, P.: The Entity-Relationship Model – Toward a Unified View of Data. ACM Transactions on Database Systems, Vol. 1, Nmbr. 1, pp. 9-36, 1976.

[ChrissisKonrad+03] Mary Beth Chrissis, Mike Konrad, and Sandy Shrum. CMMI: Guidelinesfor Process Integration and Product Improvement. Addison-Wesley Professional,Boston, MA, USA, 1st edition, February 2003. ISBN: 0321154967.

[CiaWoo01] Ciancarini, P. and Wooldridge, M. J.: Agent-Oriented Software Engineering. In: Agent-Oriented Software Engineering, Springer, Berlin, 2001

[Clermont03] Clermont, M.: A Scalable Approach to Spreadsheet Visualization. PhD thesis, University of Klagenfurt, Klagenfurt (Austria), 2003.

[CMMQuest08] CMM-Quest for CMMi 1.2: http://www.cmm-quest.com/about.htm, cited December 2008

[Codd70] Codd, E.: A relational model for large shared data banks. Communications of the ACM, Volume 13, Nmbr. 6, pp. 377-387, 1970.

[Codd82] E.Codd. Relational database: A practical foundation for productivity. Communications of the ACM, Volume 25, Nmbr.2, pp. 109-117, 1982.

[Collaxa03] Programming BPEL – Collaxa Developers Guide. Verison 2.0 Collaxa Inc-2003.

[daSilveira02] Márcio Luiz Barosso da Silveira. IT Measurement – PracticalAdvice from the Experts. Addision-Wesley Information Technology Series. Boston. MA. USA. 2002

[Daskalantonakis94] Michael K. Daskalantonakis. Achieving higher SEI levels. IEEE Software,pages 17–24, July 1994.

[DasBas+91] Michael Daskalantonakis, Victor R. Basili, and Robert Yacobellis. A method for assessing software measurement technology. Quality Engineering,3(1):27–40, 1991.

[DAML08] DAML, “DAML Ontology Library”, http://www.daml.org/ontologies/. cited: December 2008

[DawsonNolan03] 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.

[DekkersMcQuaid02] Carol A. Dekkers and Patricia A. McQuaid. The dangers of using software metrics to (mis)manage. IEEE IT Professional, pages 24–30, March 2002.

[DeMarco82] Tom DeMarco. Controlling Software Projects – Management, Measurement & Estimation. Prentice Hall PTR, Englewood Cliffs, NJ, USA, June 1982

[DeMarco89] Tom DeMarco. Software-Projektmanagement. Ulenspiegel Druck GmbH. München. 1989

[Dennis02] 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.

[Desharnais94] 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.

[Devedzic06] V. Devedzic, "Semantik Web and Education", Springer Publishing, Berlin, 2006.

[Dimitrov08] Dimitrov, Evgini: SOA in der Telekom-Branche, SOA Roundtable FHW Berlin 2008

[DITC08] Data & Analysis Center for Software (DACS), a sub-division of Defense Technical Information Center (DTIC): The Software Reliability Dataset. http://www.dacs.dtic.mil/databases/sled/swrel.shtml cited: December 2008.

[DostalJeckle04] W. Dostal, M. Jeckle. Semantik, Odem einer service-orientierten Architektur. Java Spektrum 1/2004. SIGS-DATACOM, February/March 2004

[DostalJeckle^{+}05] W. Dostal, M. Jeckle, I. Melzer, B. Zengler. Service-orientierte Architekturen mit Web Services. Akademischer Verlag, München. 2005

[Drouin95] Jean-Normand Drouin. The spice project: An overview. IEEE SoftwareProcess Newsletter, 3(2):8–9, Winter 1995.

[Dumke96] Reiner R. Dumke. CAME Tools – Lessons Learned- Proceedings of the Fourth International Symposium on Assessment of Software Tools. May 1996. Toronto, Canada.

[DumGri96] Reiner R. Dumke, H. Grigoleit. Efficiency of CAME Tools in Software Quality Assurance. Software Quality Journal. 1996

[Dumke03] Reiner R. Dumke Softwarequalitätsmanagement. http://ivs.cs.uni-magdeburg.de/~dumke/ST2/St2skript.html. Cited: Januar 2009

[Dumke03b] Reiner R. Dumke. (Editor): Software Engineering. 4th Revised and Extended Edition, Vieweg Verlagsgesellschaft mbH, Wiesbaden (Germany), 2003

[Dumke04] Reiner R. Dumke. Software Engineering- Eine Einführung für Informatiker, Ingenieure: Systeme, Erfahrungen, Methoden, Tools. Vieweg. Wiesbaden. 2004.

[Dumke05] Reiner R. Dumke. Software measurement frameworks. In Proceedingsof the 3rd World Congress for Software Quality, volume III, Online Supplement,pages 75–84, Munich, Germany, September 2005. InternationalSoftware Quality Institute (isqi), Erlangen, Germany. ISBN: 3-9809145-3-4.

[DumBra+07] Reiner R. Dumke, René Braungarten, Steffen Mencke, Karsten Richter, Hashem Yazbek. Experience-Based Software Measurement and Evaluation Considering Paradigm Evolution. Büren et al.: Metrikon 2007 – Praxis der Software-Messung, Shaker-Publ., 2007,

[DumBraKunz+05] Reiner R. Dumke, René Braungarten, Martin Kunz, and Heike Hegewald. An ISO 15939-Based Infrastructure Supporting the IT Software Measurement. In: Büren et al.: Praxis der Software-Messung – Tagungsband des DASMA Software Metrik Kongresses (MetriKon 2005), Shaker Verlag, Aachen, 2005, pp. 87-106, ISBN 3-8322-4615-0

[DumkeKunz+05a] Reiner R. Dumke, Martin Kunz, Heike Hegelwald, Hashem Yazbek. An Agent-based Measurement Infrastructure. In: A. Abran and R.R. Dumke: Proceedings of the 15th Workshop on Software Measurement (IWSM05), September 12-14, 2005, Montréal, Canada, Shaker Verlag, Aachen, pp. 79-94, ISBN 3-8322-4405-0

[DumkeKunz+05b] Dumke, R.; Kunz, M.; Riekehr, A.:Software e-Measurement in the WWW. Metrics News, Journal of the GI-Interest Group on Software Metrics, 10(1):37-42, August 2005. ISSN 1431-8008

[DumkeKunz+06a] Dumke, R. R.;Braungarten, R.; Kunz, M.; Schmietendorf, A.; Wille, C.: Strategies and Appropriateness of Software Measurement Frameworks. In: A. Abran; R. Dumke; M. Ruiz: Proceedings of the International Conference on Software Process and Product Measurement (MENSURA 2006), November, 6-8, 2006, Cádiz, Spanien, 2006, Servicio de Publicaciones de la Universidad de Cádiz, Spain, pp. 150-170, ISBN13: 978-84-9828-101-9, ISBN10: 84-9828-101-6

[DumkeKunz+08] Reiner R. Dumke, Martin Kunz, Ayaz Farooq, Konstantina Georgieva, Heike Hegewald. Formal Modelling of Software Measurement Levels of

Paradigm-Based Approaches. Technical Report. University of Magdeburg. Department of Computer Science. Magdeburg. Germany. 2008

[Ebert97] Christof Ebert. The road to maturity: Navigating between craft and science. IEEE Software, 14(6):77–82, November 1997.

[Ebert+05] Christof Ebert, Reiner R. Dumke, Manfred Bundschuh, and Andreas Schmietendorf.Best Practices in Software Measurement — How to use metricsto improve project and process performance. Springer, Berlin Heidelberg,2005.

[EbertDumke07] Christof Ebert, Reiner R. Dumke. Software Measurement. Springer Berlin 2007. ISBN 978-3-540-71648-8.

[Eclipse07] Eclipse.org: Eclipse Documentation – Latest Release Eclipse 3.3. Online Ressource,http://help.eclipse.org, Cited: 10/2007.

[EmamBriand97] K. El Emam and L. Briand, Cost and Benefits of Software Process Improvement, tech. report ISERN-97-12, International Software Engineering Research Network, 1997

[EncBrit08] Encyclopedia Britannica. 2008

[Erl05] Erl, T. “Service-Oriented Architecture Concepts, Technology, and Design” Prentice space Hall, 2005

[ErnstHage+02] Dietmar Ernst, Holger Hage, Dietrich Hofmann, Gerhard Linß. Breakthrough in Bridging the Digital Gap in Real-time e-Measurement, e-Training & e-Services. http://imeko.mit.tut.fi/vw2002/ernst.pdf cited December 2008.

[EweWag05] Ralf Ewert, Alfred Wagenhofer. Interne Unternehmensrechnung. Springer Berlin. 2005 ISBN 978-3-540-23617-7

[Farooq05] Conception and Prototypical Implementation of a Web Service as an Empirical-based Consulting about Java Technologies. Master Thesis. Otto-von-Guericke University Magdeburg. 2005

[FarBra+05] Farooq, A., Braungarten, R., Dumke, R.R.: An Empirical Analysis of Object-Oriented Metrics for Java Technologies. Proceedings of the 9th IEEE International Multi Topic Conference (INMIC2005), National University of Computer and Emerging Sciences, Karachi/Pakistan, December 2005

[FarBraKunz+06] Farooq, A.; Braungarten, R.; Kunz, M.; Schmietendorf, A.; Dumke, R. R.: Towards SOA-based approaches for IT Quality Assurance. Proceedings of the CONQUEST 2006 - Software Quality in Service-Oriented Architectures, pp. 45-54, dpunkt.verlag, Heidelberg, Germany, September 2006, ISBN-10 3-89864-432-4, ISBN-13 978-3-89864-432-7

[FarKerKunz$^+$06] Farooq, A.; Kernchen, S.; Kunz, M.; Dumke, R.; Wille, C.: Complexity and Quality Evaluation of Basic Java Technologies. In: A. Abran; M. Bundschuh; G. Büren; R.R. Dumke: Proceedings of the International Workshop on Software Measurement and DASMA Software Metrik Kongress (IWSM/MetriKon 2006), 2.-3. November 2006, Potsdam, Germany, Shaker Publ., pp. 471-482, ISBN 3-8322-5611-3

[Fenton91] Norman E. Fenton. Software Metrics: A Rigorous Approach. Kluwer Acdemic Publishers, Boston, MA, USA, April 1991.

[FentonPfleeger97] Norman E. Fenton and Shari Lawrence Pfleeger. Software Metrics — A Rigerous and Practical Approach. Internationl Thomson Computer Press,London, UK, 2nd edition, 1997.

[Fikes98] R. Fikes, "Multi-Use Ontologies", Stanford University http://www.ksl.stanford.edu/people/fikes/cs222/1998/Ontologies/sld001.htm. cited: December 2008.

[Ford93] Garry Ford. Lecture Notes on Engineering Measurement for Software Engineers. SEI educational material package. Carnegie Mellon University. USA. 1993

[Gartner07] GartnerResearch, Transforming Business, Optimize the Business Outcomes of SOA: http://h71028.www7.hp.com/enterprise/downloads/Gartner_Transforming_Business.pdf, Cited December 2008

[Goodman93] Paul Goodman. Practical Implementation of Software Metrics. McGraw-Hill International Software Quality Assurance. London .1993. ISBN 978-0077076658

[Goodman04] Paul Goodman. Software Metrics — Best Practices for Successful IT Management. Philip Jan Rothstein, FBCI, Brookfield, CT, USA, 2004. ISBN 1-931332-26-6.

[GrableJernigan$^+$99] 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.

[GresseHoisl$^+$95] 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.

[Gruber93] T. R. Gruber, “A Translation Approach to Portable Ontology Specifications”, Knowledge Acquisition, 5, pp. 199-220, 1993.

[HallFenton97] Tracy Hall and Norman Fenton. Implementing effective software metrics programs. IEEE Software, 14(2):55–64, 1997.

[Halstead77] Maurice Howard Halstead: Elements of software science. Elsevier, New York u.a. 1977, ISBN 0-444-00205-7 (Operating and programming systems series; 2).

[Hansen08] Antje Hansen. Konzeption und prototypische Realisierung eines Metriken-Cockpits für das Software-Projektmanagement. Diplomarbeit. Otto-von-Guericke Universität Magdeburg. 2008

[Härder87] Härder, T.: Realisierung von operationalen Schnittstellen. In: Lockemann, P.; Schmidt, J. (Hrsg.): Datenbank-Handbuch, S. 163-335, Springer-Verlag, Berlin (Germany), 1987.

[Harrison00] Warren Harrison. A Universal Metrics Repository. In: Proc. of the Pacific Northwest Software Quality Conference, Portland (USA), 2000.

[Harrison04] Warren Harrison. A flexible method for maintaining software metrics data: a universal metrics repository. Journal of Systems and Software, 72(2):225–234, 2004.

[HenryKafura81] Sallie Henry and Dennis Kafura. Software structure metrics based on information low. IEEE Transactions on Software Engineering, SE-7(5):510–518, September 1981.

[Hertel08] Frank Hertel. Konzeption und prototypische Realisierung einer Messtoolverwaltung für das projektbezogene Qualitätsmanagement. Diplomarbeit. Otto-von-Guericke Universität Magdeburg. 2008

[Hetzel90] Bill Hetzel. The software measurement challenge. In Proceedings of the First International Conference on Applications of Software Measurement, November 1990.

[HeuerSaake^{+}07] Heuer, A.; Saake, G., Sattler, K.U. : Datenbanken: Konzepte und Sprachen. 3rd, Updated and Enhanced Edition, MITP Publishing, Germany, 2007.

[Holmes02] 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.

[HoriEuzenat+03] Masahiro Hori, Jérôme Euzenat, Perter F. Patel-Schneider. OWL Web Ontology Language XML Presentation Syntax. W3C Consortium. http://www.w3.org/TR/owl-xmlsyntax/. Cited: December 2008

[Humphrey87] 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.

[Humphrey96] Watts S. Humphrey. Introduction to the Personal Software Process. Addison-Wesley Professional, Boston, MA, USA, December 1996. ISBN:0201548097.

[IEEE90] Computer Society IEEE. 610.12:1990 standard glossary of softwareengineering terminology, 1990.

[IFPUG99] IFPUG: Function Point Counting Practices Manual, Release 4.1. Westerville. OH. USA. 1999.

[ISBSG07] The International Software Benchmarking Standards Group (ISBSG): ISBSG Repository (R10) Field Description. http://www.isbsg.org/documents/datdskfd.doc, Warrandyte (Australia), 2007.

[ISO/IEC94] ISO/IEC 8402. ISO/IEC 8402:1994 – Quality Vocabulary. 1994.

[ISO/IEC98] ISO/IEC15504-1 Information technology- Software process assessment Part 1: Concepts and introductory guide. International Organization for Standardization, Geneva, Switzerland. 1998

[ISO/IEC00a] ISO/IEC 9000 EN ISO 2000 Qualitätsmanagementsysteme- Grundlagen und Begriffe. International Organization for Standardization, Geneva, Switzerland. 2000

[ISO/IEC00b] ISO/IEC 9001 EN ISO 2001 Qualitätsmanagmentsysteme Anforderungen. International Organization for Standardization, Geneva, Switzerland. 2000.

[ISO/IEC01] ISO/IEC. ISO/IEC 9126-1 Information Technology – Software engineering– Product quality – Part 1: Quality model. International Organization for Standardization, Geneva, Switzerland, 2001.

[ISO/IEC02] ISO/IEC. ISO/IEC 15939 — Information Technology — Software Engineering — Software Measurement Process. International Organization for Standardization, Geneva, Switzerland, 2002.

[ISO/IEC03a] ISO/IEC. ISO/IEC 9126-2 Information Technology – Software engineering– Product quality – Part 2: External Metrics. International Organization for Standardization, Geneva, Switzerland, 2003.

[ISO/IEC03b] ISO/IEC. ISO/IEC 9126-3 Information Technology – Software engineering– Product quality – Part 3: Internal Metrics. International Organization for Standardization, Geneva, Switzerland, 2003.

[ISO/IEC04] ISO/IEC. ISO/IEC 9126-4 Information Technology – Software engineering– Product quality – Part 4: Quality in Use Metrics. International Organization for Standardization, Geneva, Switzerland, 2004.

[ISO/IEC04b] ISO/IEC 90003 Software engineering- Guidelines for the application of ISO/IEC 9001:2000 to computer software. International Organization for Standardization, Geneva, Switzerland. 2004.

[ISO/IEC05] ISO/IEC. The iso survey — 2005. http://www.iso.org/iso/en/iso9000-14000/pdf/survey2005.pdf, December 2005. Geneva, Siwtzerland.

[IveMat00] 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.

[Johnson01] Philip M. Johnson: Accelerating adoption of empirically guided software development through non-disruptive, developer-centric, in-process data collection and analysis. / University of Hawaii, Department of Information & Computer Sciences. Honolulu, November 2001

[JohKou^{+}05] Philip M. Johnson; Hongbing Kou, Michael Paulding ; Qin Zhang; Aaron Kagawa, Takuya Yamashita: Improving Software Development Management through Software Project Telemetry. In: IEEE Software 22 (2005), August, Nr. 4, P. 76-85

[Jones96] Capers Jones: The economics of software process improvement. IEEEComputer, 1996.

[Jones03] Cheryl L. Jones. Implementing a successful measurement program: Tried and true practices and tools. Cutter IT Journal, 16(11):12–18, November 2003.

[KanerBond04] 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.

[KerKunz+07a] Kernchen, S.; Kunz, M.; Dumke, R.: "Proactive Class Schedule". IEEE Multidiscplinary Engineering Education Magazine, Vol. II, No. 3, 2007

[KerKunz+07b] Kernchen, S.; Kunz, M.; Dumke, R.: Proactive Class Schedule. In: Proceedings of International Conference on Computer Aided Blended Learning (ICBL2007), 7.-9. May, 2007, Florianopolis, Brazil.

[KerKunz+07c] Kernchen, S.; Kunz, M.; Schmietendorf, A.; Dumke, R.: Ontology Metrics- Measurement and Evaluation of Ontologies. In: T. Dekkers. Proceedings of the 4th Software Measurement European Forum (Smef 2007), 9.-11. May,2007 Rome, Italy.

[KernchenRud+07] Steffen Kernchen, Dmytro Rud, Fritz Zbrog, and Reiner R. Dumke, “Processing Remote Measurement Databases by the Means of Mobile Agents”, In Proceedings of the 3rd International Conference on Web Information Systems and Technologies, Barcelona, Spain, March 2007.

[King04] King, P.: SLED – Defense Technical Information Center, U.S. D.o.D., Fort Belvoir (USA), 2004

[Kitchenham96] Barbara A. Kitchenham. Desmet: A method for evaluating software engineeringmethods and tools. Technical Report TR96-09, Department ofComputer Science, University of Keele, Keele, Staffordshire, UK, August1996.

[Kitchenham07] Kitchenham, B.: Empirical Paradigm – The Role of Experiments. In: Basili et al.: Emiprical Software Engineering, Springer-Publishing, 2007

[KnapikJohnson98] Knapik, M. and Johnson, J.: Developing Intelligent Agents for Distributed Systems, McGraw-Hill, New York, 1998

[KnöllBusse91] H.D. Knöll, J. Busse. Aufwandsschätzung von Softwareprojekten in der Praxis. BI Wissenschaftsverlag. Mannheim. 1991

[KnuDim+03] M. Knuth (Editor), Evgini Dimitrov, M. Großmann, J. Lezius, G. Pöhner, Daniel Reitz, J. Renner, M. Schmidt, Andreas Schmietendorf. Web Services- Einführung und Übersicht. Software&Support Publishing. Frankfurt 2003

[KochKuvaja+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.

[KosLey04] D. Kossmann, F. Leymann. Web Services. Informatik Spektrum. Band 27. Volume 2. Springer Publishing Heidelberg. 2004

[Kriz88] Jürgen Kriz, editor. Facts and Artefacts in Social Science: An Ephistemological and Methodological Analysis of Empirical Social Science. McGraw HillResearch, New York, NY, USA, 1988.

[Kueng00] Peter Kueng. Process performance measurement system — a tool to supportprocess-based organizations. Total Quality Management, 11(1):67–85, January 2000.

[KugRem95] Hans-Jürgen Kugler and Santiago Rementeria. Software engineeringtrends in europe. Technical report, European Software Institute (ESI),Bilbao, Spain, 1995.

[Kunz+05] Kunz, M.; Braungarten, R.; Dumke, R.R.: Measuring eLearning - A classification approach for eLearning Systems. In: A. Abran and R.R. Dumke: Proceedings of the 15th Workshop on Software Measurement (IWSM05), September 12-14, 2005, Montréal, Canada, Shaker Verlag, Aachen, pp. 79-94, ISBN 3-8322-4405-0

[Kunz+06a] Kunz, M.; Schmietendorf, A.; Braungarten, R.; Dumke, R.: Serviceorientierte Ausrichtung von Test- und Messwerkzeugen. In: A. Schmietendorf and R. Dumke: Tagungsband 1. Workshop Bewertungsaspekte serviceorientierter Architekturen (BSOA06), 24. November, 2006, FHW Berlin, Germany, Eigenverlag Otto-von-Guericke-University Magdeburg, pp. 11-20, ISBN 3-929757-95-8

[Kunz+06b] Kunz, M.; Schmietendorf, A.; Dumke, R.; Rud, D.: SOA-Capability of Software Measurement Tools. In: A. Abran; R. Dumke; M. Ruiz: Proceedings of the International Conference on Software Process and Product Measurement (MENSURA 2006), November, 6-8, 2006, Cádiz, Spanien, 2006, Servicio de Publicaciones de la Universidad de Cádiz, Spain, pp. 216-225, ISBN13: 978-84-9828-101-9, ISBN10: 84-9828-101-6

[Kunz+06c] Kunz, M.; Kernchen, S.; Dumke, R.; Schmietendorf, A.: Ontology-based web-service for object-oriented metrics. In: A. Abran; M. Bundschuh; G. Büren; R.R. Dumke: Proceedings of the International Workshop on Software Measurement and DASMA Software Metrik Kongress (IWSM/MetriKon 2006), 2.-3. November 2006, Potsdam, Germany, Shaker Publ., pp. 99-106, ISBN 3-8322-5611-3

[Kunz+06d] Kunz, M.; Leszak, M.; Braungarten, R.; Dumke, R.R.: Design of an Integrated Measurement Database for Telecom Systems Development.

In: A. Abran; M. Bundschuh; G. Büren; R.R. Dumke: Proceedings of the International Workshop on Software Measurement and DASMA Software Metrik Kongress (IWSM/MetriKon 2006), 2.-3. November 2006, Potsdam, Germany, Shaker Publ., pp. 471-482, ISBN 3-8322-5611-3

[Kunz+06e] Kunz, M.; Braungarten, R.; Dumke, R.: Bewertungsansätze und Modelle für unternehmensweite Software-Messinitiativen. Arbeitskonferenz Softwarequalität und Test (ASQT) 2006, Klagenfurt, Austria, 14.-15. September 2006

[Kunz+06f] Kunz, M.; Schmietendorf, A.; Dumke, R.; Wille, C.: Towards a service-oriented measurement infrastructure. In: T.Dekkers. Proceedings of the 3rd Software Measurement European Forum (Smef 2006),10.-12. May 2006, Rome, Italy.

[Kunz+07a] Kunz, M.; Dumke, R.; Braungarten, R.; Schmietendorf, A.: How to measure Agile Software Development. Proceedings of the International Conference on Software Process and Product Measurement (IWSM-Mensura 2007), 5.-8. November 2007, Palma de Mallorca, Spain, pp. 319-325, ISBN 978-84-8384-020-7

[Kunz+07b] Kunz, M.; Dumke, R.: Empirische Grundlagen zur COSMIC-FFP-Anwendung für die Aufwandsschätzung Preprint Nr. 7, Computer Science Departement, University Magdeburg, 2007

[Kunz+08a] Kunz, M.; Mencke, S.;Zenker, N.; Braungarten, R.; Dumke, R.; Quality-driven orchestration of services; IWSM-Mensura 2008, Lecture notes in computer science ; 5338 Software process and product measurement . - Berlin : Springer, ISBN 3-540-89402-0. – 2008

[Kunz+08b] Kunz, M.; Mencke, S.;Zenker, N.; Rud, D.; Dumke, R.; Empirical based, quality-driven orchestration of services: BSOA 2008 . - Aachen : Shaker, ISBN 978-3-8322-7221-0. - 2008

[Kunz+08c] Kunz, M.; Mencke, S.; Rud, D.; Dumke, R.; Empirical-Based Design - Quality-Driven Assembly of Components. In Proceedings of the 2008 IEEE International Conference on Information Reuse and Integration (IEEE IRI-2008), Las Vegas, USA 13-15. July 2008, ISBN 978-1-4244-2659-1

[Kunz+08d] Kunz, M.; Zenker, N.; Mencke, S.; Dumke, R.; "Unit Metrics" - A Tool to support Refactoring in Agile Software Development. In Proceedings of the 2008 World Congress in Computer Science Computer Engineering and Applied Computing (Worldcomp´08), Las Vegas, USA 14-17. July 2008, ISBN 1-60132-090-6

[Kunz$^+$08e] Kunz, M.; Dumke, R.; Zenker, N.; Software Metrics for Agile Software Development. In Proceedings of the 19th Australien Software Engineering Conference (ASWEC 2008); Perth, Australia 25-28. March 2008; IEEE Computer Society, ISBN 978-0-7695-3100-7

[LairdBrennan06] Laird, L. M.; Brennan, M. C.: Software Measurement and Estimation – A Practical Approach. IEEEComputer Science, 2006

[LeeHendler$^+$01] T.B. Lee, J. Hendler, and O. Lassila, "The Semantic Web", Scientific American, 284, pp. 34-44, 2001.

[ListBruckner$^+$05] Beate List, Robert M. Bruckner, and Jochen Kapaun. Holistic softwareprocess performance measurement from the stakeholders' perspective. In Proceedings of the 16th International Workshop on Database and ExpertSystems Applications (DEXA'05), pages 941–947. Vienna University of Technology, Austria, IEEE Computer Society, August 2005.

[Lonchamp93] 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.

[LotBraKunz$^+$05] Mathias Lother, René Braungarten, Martin Kunz, and Reiner R. Dumke. The Functional Size eMeasurement Portal (FSeMP) - A Web-based Approach for Effort Estimation, Benchmarking and eLearning. In: Abran et al.: Software Measurement - Research and Application, Proc. of the IWSM/Metrikon 2004, Berlin, Germany, November 2004, Shaker Publ., pp. 27-40, ISBN 3-8322-3383-0

[LoDuBrKunz$^+$05] Lother, M.; Dumke, R.; Braungarten, R.; Kunz, M.: Ein Portal zur funktionalen Größenmessung von Software. Softwaretechnik Trends, 25 (2005) 1, pp. 39-44, ISSN 0720-8928

[LotSchm$^+$02] Mathias Lother, Andreas Schmietendorf,T. Böhm, Reiner R. Dumke. Quality Evaluation of Large-scale Software Systems. In R.R. Dumke et al (Editors). Software Measurement and Evaluation- Proceedings of the 12th International Workshop on Software Measuremen. Shaker Publishing. Aachen 2002.

[LotherBöhm02] Mathias Lother, T. Böhm. Qport- Quality Evaluation of Telecommunication Systems- Project internal Report. Magdeburg. 2002

[Lother07] Mathias Lother. From Software Measurement to e-Measurement. Shaker Publishing. Aachen. 2007

[MacKenLas+06] C. MacKenzie, K. Laskey, F. McCabe, P.F. Brown, R. Metz. Reference Model for Service Oriented Architectur 1.0 OASIS, July 2006

[MatthesSchmidt99] Matthes, F.; Schmidt, J.W.: Datenbanken und Informationssysteme. Lecture Notes, TU Harburg, http://www.sts.tu-harburg.de/teaching/ws-98.99/DBIS/91-dbis.pdf, Hamburg (Germany), 1999

[McCabe76] Thomas J. McCabe. A complexity measure. IEEE Transactions on Software Engineering, SE-2(4):308–320, December 1976.

[McCallRichards+77] J.A. McCall, P.K. Richards, G.F. Walters. Factors in Software Quality. Volume I: Concepts and Definitions of Software Quality. Technical Report. RADC-TR-77-369. Rome Air Development Center. Griffiss Air Force Base. USA. 1977.

[McClure92] McClure, C.: The Three R's of Software Automation: Re-Engineering, Repository, Reusability. Prentice Hall, London (UK), 1992.

[McInnis99] McInnis, K.: Managing The Microsoft Repository AD131. http://www.cbd-hq.com/PDFs/repository.pdf, Dallas (Texas), 1999.

[McQuaidDekkers04] 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.

[MIT05] MIT (Massachusetts Institute of Technology) Kerberos: The Network Authentication Protocol, http://web.mit.edu/kerberos/ 2005

[Meinel06] A. Meinel. Serviceorientierung verändert Paradigmen – Im Gespräch. ComputerZeitung 28. July 2006

[MenDub07] Menascé, D. A. and Dubey, V.: Utility-based QoS Brokering in Service Oriented Architectures. In Proceedings of the IEEE International Conference on Web Services (ICWS 2007)

[Mencke07] Mencke, Steffen, Dumke, Reiner R., Agent-Supported e-Learning. Preprint No 8, Department of Computer Science, University of Magdeburg, 2007

[Mencke08] Proactice Ontology-Based Content Provision in the Context of e-Learning. PhD thesis, Otto-von-Guericke University of Magdeburg, 2008.

[MenckeKunz^{+}08a] Mencke, S.; Kunz, M.; Zenker, N.; Dumke, R.; Ontology-Based Generic Learning Path Recommendations. In Proceedings of the 2008 World Congress in Computer Science Computer Engineering and Applied Computing (Worldcomp´08), Las Vegas, USA 14-17. July 2008, ISBN 1-60132-090-6

[MenckeKunz^{+}08b] Mencke, S.; Kunz, M.; Dumke, R.; Towards Metrics for Ontology Balance. In Proceedings of the 20th International Conference on Software Engineering & Knowledge Engineering (SEKE 2008), San Francisco, USA 1-3. July 2008, ISBN 1-891706-22-5

[MenckeKunz^{+}08c] Mencke, S.; Kunz, M.; Pukal, M.; "Runtime Adaptations within the QuaD^{2}-Framework", In Proceedings of the 5th ECOOP'2008 Workshop on Reflection, AOP and Meta-Data for Software Evolution (RAM-SE 2008), Paphos, Cyprus, July 7, 2008.

[MenckeKunz^{+}08d] Mencke, S.; Kunz, M.; Dumke, R.; Steps to an Empirical Analysis of the Proactive Class Schedule. In Proceedings of Interactive Mobile and Computer Aided Learning Conference, IMCL 2008; Amman, Jordan 16-18. April 2008

[Mendonça97] Manoel Gomes Mendonça. An Approach to Improving Existing MeasurementFrameworks in Software Development Organizations. PhD thesis,University of Maryland, 1997.

[Microsoft08] Microsoft Corporation. Using the Microsoft Repository. http://msdn.microsoft.com/en-us/library/aa224785(SQL.80).aspx
Cited: December 2008.

[Minkiewicz00] Arlene Minkiewicz. Software measurement? what’s in it for me? In Proceedings of the SM /ASM 2000 Conference, 2000.

[MoraDenger03] Maricel Medina Mora and Christian Denger. Requirement metrics. an initialliterature survey on measurement approaches for requirement specifications.Technical Report 096.03/E, Fraunhofer IESE, Kaiserslautern,Germany, October 2003.

[Natis03] Y.V. Natis. Service-Oriented Architecture Scenario. ID Number AV-19-6751. Gartner Research. April 2003

[Naumann05] Naumann, F.: Mediator/Wrapper: Architektur & Peer-Data-Management. Information Integration. http://www.informatik.hu-berlin.de/forschung/gebiete/wbi/ii/folien/InfoInt_07_MediatorWrapperPDMS.ppt. Cited: December 2008. Berlin 2005

[NewLom03] Newcomer, E., Lomow, G. "Understanding SOA with Web Services" Addision Wesley, 2003

[ODMG97] Cattell, R.G.G. et al. (Editor): The Object Database Standard ODMG 2.0. Morgan Kaufmann Publishers, San Francisco (USA), 1997.

[OmanPfleeger96] Paul Oman and Shari Lawrence Pfleeger, editors. Applying Software Metrics. Wiley-IEEE Computer Society Press, New York, NY, USA, first edition, 1996.

[OMG06] Business Process Modeling Notation (BPMN) Specification. http://www.omg.org/docs/dtc/06-02-01.pdf. Version: February 2006

[Pall87] Gabriel A. Pall. Quality Process Management. Prentice Hall, Englewood Cliffs, NJ, USA, 1st edition, May 1987.

[PanaitLuke06] Panait, L. and Luke, S.: Selecting Informative Actions Improves Cooperative Multiagent Learning. In: Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems, AAMAS 200, Hakodate, Japan, May 8-12, pp. 760-766, 2006

[Pandian03] C. Ravindranath Pandian. Software Metrics. Auerbach Publications, NewYork, NY, USA, 2003.

[Pandian04] Pandian, C. R.: Software Metrics – A Guide to Planning, Analysis, and Application. CRC Press Company, 2004

[Peltz03] Peltz, C. "Web Services Orchestration and Choreography", Computer, 2003

[PerkinsPeterson+03] Tim Perkins, Ronald Peterson, and Larry Smith. Back to the basics: Measurementand metrics. STSC CrossTalk, pages 9–12, December 2003.

[Pfleeger93] Shari Lawrence Pfleeger. Lessons learned in building a corporate metrics program. IEEE Software, 10(3):67–74, May 1993.

[Pfleeger97] Shari Lawrence Pfleeger. Assessing measurement. IEEE Software, 14(2):25–26, March/April 1997.

[Plenum06] plenum, Management Consulting G.: IT-Cockpit – Kennzahlenbasierte Steuerung von Kosten und Nutzen Ihrer IT. 2006

[PricCoo04] PricewaterhouseCoopers: The Use of Spreadsheets: Considerations for Section 404 of the Sarbanes-Oxley Act. http://www.pwc.com/extweb/service.nsf/docid/CD287E403C0AEB7185256F08007F8CAA, New York (USA), 2004.

[powerSoftware08] powerSoftware: Essential Metrics, Krakatau Professional http://www.powersoftware.com/em/, Cited December 2008

[PSM01] McGarry, J.; Card, D.; Jones, C.; Layman, B.; Clark, E.; Dean, J. & Hall, F. “Practical space Software Measurement: Objective Information for Decision Makers” Addison-Wesley, 2001

[Rational08] IBM Rational Software: Project Console, http://www.gsi.com.ar/brochures/projectconsole.pdf, cited December 2008

[Reifer02] D. Reifer, “Let the Numbers Do the Talking,” CrossTalk, Mar. 2002, pp. 4–8.

[ReiSchm^{+}03] Daniel Reitz, Andreas Schmietendorf, Reiner R. Dumke, Evgeni Dimitrov. J. Lezius, T. Schlosser. Aspekte des empirischen Software Engineerings im Umfeld von Enterprise Application Integration Lösungen. Preprint Nr. 5. Otto-von-Guericke Universität Magdeburg. 2003.

[Richter05] Richter, K.: Softwaregrößenmessung im Kontext von Software-Prozessbewertungsmodellen. Diploma Thesis, University of Magdeburg, 2005

[Rico04] D.F. Rico, ROI of Software Process Improvement: Metrics for Project Managers and Software Engineers, J. Ross Publishing, 2004.

[Rubin87] 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.

[RoweWright01] Rowe and Wright. Expert Opinions in Forecasting. Role of the Delphi Technique. In: Armstrong (Ed.): Principles of Forecasting: A Handbook of Researchers and Practitioners. Kluwer Academic Publishers. Boston, 2001.

[Rubin90] Howard A. Rubin. Measurement — where we’ve been. The Rubin Review, 3(3), July 1990.

[RudKunz^{+}07] Rud, D.; Kunz, M.; Schmietendorf, A.; Dumke, R.: Performance Analysis in WS-BPEL-Based Infrastructures. In: Proceedings of the 23rd Annual UK Performance Engineering Workshop (UKPEW2007), pp. 130-141, July 2007, Edge Hill University, Ormskirk, Lancashire, UK

[RuSchmKunz^{+}07a] Rud, D.; Schmietendorf, A.; Kunz, M.; Dumke, R.R.: Prozessqualität bei dem Übergang zur serviceorientierten Architektur. In: G. Büren; M. Bundschuh; R.R. Dumke: Praxis der Software-Messung - Tagungsband des DASMA Software Metrik Kongress (MetriKon 2007), 15.-16. November 2007, Kaiserslautern, Shaker Verlag, pp. 141-154, ISBN 978-3-8322-6703-2

[RuSchmKunz$^+$07b] Rud, D.; Schmietendorf, A.; Kunz, M.; Dumke, R.: Analyse verfügbarer SOA-Reifegradmodelle - State-of-the-Art.. In A.Schmietendorf; M. Mevius; R.R. Dumke: Tagunsband 2. Workshop Bewertungsaspekte serviceorientierter Architekturen (BSOA 2007), pp. 115-126, November 2007, Karlsruhe, Germany, Shaker Pupl. ISBN 978-3-8322-6716-2

[RumPre$^+$94] Rumbaugh, J.; Blaha, M.; Premerlani, W.: Eddy, F.; Lorensen, W.: Objektorientiertes Modellieren und Entwerfen. Carl Hanser Verlag, Munich (Germany), 1994.

[Russac02] Janet Russac. IT Measurement: Practical Advice from the Experts, chapter 18—Cheaper, Better, Faster: A Measurement Program ThatWorks, pages 147–158. 1. Addison-Wesley, Boston, MA, USA, 2002.

[SciToo08] Scientific Toolworks: Understand SourceCode Analysis and Metrics, http://www.scitools.com/products/understand/, Cited December 2008

[Schema08] SchemaWeb, “SchemaWeb”, http://www.schemaweb.info/. Cited: December 2008.

[Schmietendorf07] Andreas Schmietendorf. Eine strategische Vorgehensweise zur erfolgreichen Implementierung serviceorientierter Architekturen in großen IT Organisationen. Shaker Publishing Aachen. 2007

[SchmKunz$^+$07] Schmietendorf, A.; Kunz, M.; Dumke, R.: Empirical analyses about the granularity of industrially used Web Services. In: Proceedings of the 10th International Conference on Quality Engineering in Software Technology (CONQUEST2007), September 2007.

[SchmiDim04] Andreas Schmietendorf, Evgini Dimitrov. “Implementation of services-oriented infrastructures by the use of a Service Centre” Proc. of the 2nd International Workshop on eServices and eLearning, Plovdiv 2004

[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.

[Senko73] Senko, M.: Data structures and access in database systems. IBM Systems Journal, Volume 12, pp. 30-93, 1973.

[ShanHua06] T.C. Shan, W. Hua. Solution Architecure for N-Tier Applications. In proceedings IEEE International Conference on Service Computing. IEEE Computer Society, Los Alamitos. CA. USA. September 2006

[Shaw90] Mary Shaw. Prospects for an engineering discipline of software. IEEE Software 7 (6):15-24. 1990

[Short02] Scott Short. Building XML Web Services for the Microsoft .Net Platform. Microsoft Press. 2002

[SirinParsia+05] Sirin,E., Parsia B. and Hendler, J.: Template-based composition of semantic web services. In Proc. AAAI fall symposium on agents and the semantic web, Virginia, USA (2005)

[Smlab09] SML@b Web Site: www.smlab.de (2009), Cited: Januar 2009

[Sneed05] Harry Sneed. Software-Projektkalkulation. Hanser Publishing, Berlin 2005

[Solingen04] Rini van Solingen. Measuring the ROI of Software Process Improvement. IEEE Software May/June 2004.

[SolBer99] Rini van Solingen, E. Berghout. The Goal/Question/Metric Method. McGraw-Hill. London. 1999

[SPC04] Software Productivity Center Inc.: Creating a Metrics Program, Step 7: Create a Metrics Database. Vancouver (Canada), 2004.

[Stein04] Stein, B.: Einführung in Datenbanken. Lecture notes, University of Weimar, Weimar (Germany), 2004.

[Telelogic08] IBM Telelogic Dashboard, http://www.telelogic.com/products/dashboard/index.cfm, Cited December 2008

[TsiKlu78] Tsichritzis, D.; Klug, A.: The ANSI/X3/SPARC DBMS Framework Report of the Study. Group on Database Management Systems. Information Systems 3, pp. 173-191, 1978.

[Ullwer06] Christof Ullwer. Konzeption und prototypische Realisierung einer Telemetrie-basierten Mess-Architektur. Diplomarbeit. Otto-von-Guericke Universität Magdeburg. 2006.

[UmaEmu05] 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.

[Vossen00] Vossen, G.: Datenmodelle, Datenbanksprachen und Datenbankmanagementsysteme. Oldenbourg Verlag, Munich (Germany), 2000.

[Wallmüller01] Ernest Wallmüller. Softwarequalitätsmanagement in der Praxis. Hanser Verlag. München. 2001

[WangDorling+99] Yingxu Wang, Alec Dorling, Graham King, Margaret Ross, Jeff Staples,and Ian Court. A worldwide survey on best practices toward softwareengineering process excellence. ASQ Journal of Software Quality Professional,2(1):34–43, December 1999.

[Weinberg92] Gerald M. Weinberg. Quality Software Management: Systems Thinking, Volume 1. Dorset House Publishing Company, New York, NY, USA, 1992. ISBN: 0-932633-22-6.

[Weise06] Eric Weise. Konzeption und prototypische Realisierung einer Web Service-basierten Ontologie objektorientierter Metriken. Diplomarbeit. Otto-von-Guericke Universität Magdeburg. 2006.

[Whitmire97] Scott A. Whitmire. Object-Oriented Design Measurement. John Wiley &Sons, Inc., New York, NY, USA, 1st edition, 1997.

[Wiegers97] Karl E. Wiegers. Software metrics: Ten traps to avoid. Software Development, 5(10), October 1997.

[Wiegers99] Karl E. Wiegers. A software metrics primer. Software Development, July 1999.

[Wille05] Cornelius Wille, Software Agent Measurement Framework. Shaker-Publishing, Aachen 2005.

[Woods04] D. Woods. Enterprise Service Architecture – SAP's Bauplan für Geschäftsapplikationen der nächsten Generation. SAP Press. Galileo Press. Bonn. 2004.

[Wu05] Wu, T.: EII-ETL-EAI: What, Why, and How!. Information Integrator Advocate. Software Group. IBM Taiwan. https://www6.software.ibm.com/developerworks/tw/events/20051028/db2_6b.pdf. cited: December 2008.

[Zelkowitz07] Zelkowitz, M., V.: Techniques for Empircal Validation. In: Basili et al.: Empirical Software Engineering, Springer-Publishing., 2007

[ZenkerKunz$^+$07] Zenker, N.; Kunz, M.; Rautenstrauch, C.: Service Oriented Architecture: Resource Based Evaluation of a SOA . In A.Schmietendorf; M. Mevius; R.R. Dumke: Tagunsband 2. Workshop Bewertungsaspekte serviceorientierter Architekturen (BSOA 2007), pp. 23-32, November 2007, Karlsruhe, Germany, Shaker Pupl. ISBN 978-3-8322-6716-2

[ZenkerKunz$^+$08a] Zenker, N.; Kunz, M.; Mencke, S.; Resource Consumption in Heterogeneous Environments. In Proceedings of the 2008 World Congress in Computer Science Computer Engineering and Applied Computing (Worldcomp´08), Las Vegas, USA 14-17. July 2008, ISBN 1-60132-090-6

[ZenkerKunz$^+$08b] Zenker, N.; Kunz, M.; Rajub, J; Software Metrics for Educational Software Development. In Proceedings of Interactive Mobile and Computer Aided Learning Conference, IMCL 2008; Amman, Jordan 16-18. April 2008

[Zuse98] Horst Zuse. A Framework of Software Measurement. Walter de Gruyter &Co., Berlin, Germany, 1998.

Appendix A: ISO/IEC 15939 Measurement Process Activities

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.

Appendix B: Attributes of analyzed Measurement Databases

Attributes of the ARF Dataset [Harrison 2000]

Software Component Details	
Component Code	**Total Statements in Segment**
Number of Comments in Segment	**Number of Pre-Process Statements in Segment**
Subjective Complexity *(Easy, Moderate, Hard)*	**Function** *(Computational, Control, Data Accessing, Error Handling, Initialization)*
Software Problem Report Details	
Form Number	**Project Code**
Programmer Code	**Form Date**
Number of Components Changed	**Number of Components Examined**
More than one Component Affected	**Date Change was Determined**
Date Change was Started	**Code of Changed Component**
Effort for Change *(Less than 1 hour, 1 hour to 1 day, 1 day to 3 days, more than 3 days, unknown)*	**Type of Change** *(Error Correction, Planned Enhancement, Implement Requirements Change, Improve Clarity, Improve User Service, Develop Utility, Optimization, Adapt to Environment, Other)*
Implementation Technique *(The implementation techniques used on the software project expressed as a percentage of the DSLOC built using specific techniques of Structured Coding, Top Down Design and Programming, Chief Programmer Teams, Code Reviews or Inspections, and Librarian or Program Support Library.)*	
Productivity *(Ratio consisting of DSLOC to Total man-months)*	**Average Number of Personnel** *(Ratio consisting of total man-months to total months)*
Error Rate *(Ratio Consisting of Errors to DSLOC)*	

Attributes of the DACS Productivity Dataset [Harrison 2000]

Software Component Details	
Project Size *(Number of delivered source lines of code (DSLOC) in the delivered project.)*	**Project Effort** *(Effort in man months required to produce the software product.)*
Project Duration *(Duration of project in total months derived from start and end dates of projects, less any "dead time" in the project.)*	**Source Language** *(Programming languages used on the project recorded by name and expressed as a percentage of the total DSLOC written in each different language.)*
Errors *(The number of formally recorded Software Problem Reports (SPRs) for which a fix has been generated during the period covered by the project.)*	**Documentation** *(Delivered pages of documentation including program listings, flow charts (low and high level), operating procedures, etc.)*
Implementation Technique *(Percentage of DSLOC built using specific techniques of Structured Coding, Top Down Design and Programming, Chief Programmer Teams, Code Reviews and Librarian or Program Support Library.)*	**Error Rate** *(Number of formally recorded Software Problem Reports for which a fix has been generated during the period covered by the project.)*
Average Number of Personnel *(Ratio of Total man-months to Total Months)*	**Productivity** *(Ratio of DSLOC to Total man-months.)*

Attributes of the NASA Software Reliability Dataset DTIC 2004

Software Component Details	
Project Identification (System Code) *(An internally assigned identification number.)*	**Failure Number** *(A number identifying a particular failure. Failure a consecutively numbered from the first failure recorded.)*
Failure Interval *(The time elapsed from the previous failure to the current failure. For project 6, this time is given in CPU seconds; for the remaining projects, the time is given in wall-clock seconds.)*	**Day of Failure** *(Represents the day on which the failure occurred in terms of the number of working days from the start of the current phase or data collection period.)*

Attributes of the ISBSG Benchmarking Data CD Release 10 [ISBSG 2007]

Project Data Parameters

Project ID

(A primary key, for identifying projects.)

Count Approach

(A description of the technique used to count the function points; e.g. IFPUG, MKII, NESMA, COSMIC-FFP etc.)

Function Points

(The adjusted function point count number. Adjusted by the Value Adjustment Factor.)

Function Size Metric Used

(The functional size metric used to record the size of the project, e.g.. IFPUG3, IFPUG4, in-house etc.)

Value Adjustment Factor

(The adjustment to the function points, applied by the project submitter, that takes into account various technical and quality characteristics e.g.: data communications, end user efficiency etc. This data is not reported for some projects, (i.e. it equals 1).)

Counting Technique

(The technology used to support the counting process. Certain technologies used in function point counting can impact on the count's potential accuracy.)

Development Platform

(Defines the primary development platform, (as determined by the operating system used). Each project is classified as either, a PC, Mid Range or Mainframe.)

Summary Work Effort

(Provides the total effort in hours recorded against the project by the development organization. The three methods provided for are A, B and C.)

Resource Level

(Data is collected about the people whose time is included in the work effort data reported. Four levels (1 to 4) are identified in the data collection instrument.)

Data Quality Rating

(This field contains an ISBSG rating code of A, B, C or D applied to the project data by the ISBSG quality reviewers.)

Max Team Size

(The maximum number of people that worked at any time on the project, (peak team size).)

Development Type

(This field describes whether the development was a new development, enhancement or re-development.)

Reference Table Approach

(This describes the approach used to handle counting of tables of code or reference data, (a comment field).)

Architecture

(Defines the architecture type of the project. e.g.: Client/Server, LAN, WAN etc.)

Language Type

Defines the language type used for the project: e.g. 3GL, 4GL, Application Generator etc.

Primary Programming Language

The primary language used for the development: JAVA, C++, PL/1, Natural, Cobol etc.

DBMS Used

(Whether the project used a DBMS.)

Upper CASE Used

(Whether project used upper CASE tool.)

Lower CASE Used (with code generator)

(Whether project used lower CASE tool with code generator.)

Integrated CASE Used

(Whether project used integrated CASE tool.)

Used Methodology

Project Elapsed Time

(States whether a methodology was used.)

Development Techniques

(Techniques used during development. (e.g.: JAD, Data Modeling, OO Analysis etc.).)

Project Inactive Time

(This is the number of months in which no activity occurred, (e.g. awaiting client sign off, awaiting acceptance test data). This time, subtracted from Project Elapsed Time, derives the elapsed time spent working on the project.)

Defects Delivered

(Defects reported in the first month of system use. Three columns in the data covering the number of Extreme, Major and Minor defects reported.)

User Base – Locations

(Number of physical locations being serviced/supported by the installed system.)

Organization Type

(This identifies the type of organization that submitted the project. (e.g.: Banking, Manufacturing, and Retail).)

Application Type

(This identifies the type of application being addressed by the project. (e.g.: information system, transaction/production system, process control.))

Degree of Customization

(If the project was based on an existing package, this field provides comments on how much customization was involved.)

Work Effort Breakdown

(When provided in the submission, these fields contain the breakdown of the work effort reported by five categories: Plan, Specify, Build, Test and Implement.)

Percentage of Uncollected Work Effort

(The percentage of Work Effort not reflected in the reported data. i.e. an estimate of the work effort time not collected by the method used.)

Enhancement Data

(Total elapsed time for project in months.)

How Methodology Acquired

(Describes whether the methodology was purchased or developed in-house.)

Implementation Date

(Actual date of implementation. (Note: the date is shown in the data in date format 1/mm/yy).)

User Base – Business Units

(Number of business units that the system services, (or project business stakeholders).)

User Base – Concurrent Users

(Number of users using the system concurrently.)

Business Area Type

(This identifies the type of business area being addressed by the project where this is different to the organization type. (e.g.: Manufacturing, Personnel, and Finance).)

Package Customization

(This indicates whether the project was a package customization. (Yes or No).)

Project Scope

(This data indicates what tasks were included in the project work effort data recorded. These are: Planning, Specify, Design, Build, Test, and Implement.)

Ratio of Project Work Effort to Non-Project Activity

(The ratio of Project Work Effort to Non-Project Activities.)

Function Point Categories

(When provided in the submission, the following five fields which breakdown the Function Count are provided: external Inputs, external Outputs, external Enquiries, internal logical files, and external interface files.)

Total Defects Delivered

(When provided in the submission, for enhancement projects the three fields Additions, Changes, and Deletions, which breakdown the Function Point Count are provided.)

Source Lines of Code (SLOC)

(A count of the SLOC produced by the project.)

Normalized Work Effort

(For projects covering less than a full development life-cycle, this value is an estimate of the full development life-cycle effort. For projects covering the full development life-cycle, and projects where development life-cycle coverage is not known, this value is the same as Summary Work Effort.)

Unadjusted Function Point Rating

(This field contains an ISBSG rating code of A, B, C or D applied to the unadjusted function point count data by the ISBSG quality reviewers.)

(Defects reported in the first month of system use. This column shows the total of Extreme, Major and Minor defects reported. Where no breakdown is available, the single value is shown here.)

Unadjusted Function Points

(The unadjusted function point count (before any adjustment by a Value Adjustment Factor if used).)

Work Effort Unphased

(Where no phase breakdown is provided in the submission, this field contains the same value as the Summary Work Effort. Where phase breakdown is provided in the submission, and the sum of that breakdown does not equal the Summary Work Effort, the difference is shown here.)

Productivity Rates Parameters

Project ID

(The primary key, for identifying projects.)

Project Productivity Rate

(Project productivity delivery rate in hours per function point calculated from Summary Work Effort divided by Unadjusted Function Point count.)

Normalized Productivity Delivery Rate

(Project productivity delivery rate in hours per function point calculated from Normalized Work Effort divided by Unadjusted Function Point count. Use of normalized effort and unadjusted count should render more comparable rates.)

Normalized Productivity Delivery Rate (adjusted)

(Project productivity delivery rate in hours per function point calculated from Normalized Work Effort divided by Adjusted Function Point count.)

Reported Productivity Delivery Rate (adjusted)

(Project productivity delivery rate in hours per function point calculated from Summary Work Effort divided by Adjusted Function Point count.)

Appendix C: Assessment results for Measurement Tools

Measurement tool	Non exportable report	Non exportable charts	Results presented in HTML	File (Format)	DB (Interface (JDBC, …))	XML	Web-Service-oriented
JMetrics	x	x	*(planned: XML to HTML Reports)	x (Textfile)	-	*(planned)	-
Imagix 4D	x	x	x	x - RTF, -CSV [Microsoft Visio-Support]) - PS - PNG -Table (ASCII text file)	-	-	-
SLIM-Metrics and SLIM-DataManager	x	x	-	-	x (DB, ODBC)	-	-
Scientific Toolworks, Inc. **Understand for ADA** **Understand for C++** **Understand for Delphi** **Understand for FORTRAN** **Understand for Java** **Understand for JOVIAL**	x	x	x	x (PERL API, C/C++ API)	-	-	-
Function Point Workbench	x	x	x	x (spreadsheet format)	-	x (XML-File)	-
TAU / Logiscope	x	x	x	x (Word, Framemaker, Interleaf)		??	
SCOPE - Project Sizing Software	x	x	-	x (Word, Excel, PDF)	x (MsACCESS)	-	-
SDMetrics	x	x	x	x (text [tab separated tables], Openoffice.org CALC, XML for Microsoft Excel)	-	x (XML for Microsoft Excel)	-
CMM-Quest	x	x	x	x (text)	-	-	-
CMTJava, CMT++	x	x	x	x (text, Excel, XML)	-	x	-
COSTAR	x	x	-	x (text (Costar's native file format), CSV, Excel, BMP)	-	-	-
EPM – Essential Project Manager	x	x	x	x (CSV)	* (SQL DB, JDBC)	x	-
EM – Essential Metrics	x	x	x	x (CSV)	* (SQL DB,	x	-

					JDBC)		
Krakatau Essential Metrics	x	x	x	x (CSV)	x (MySQL data file)	x	-
Krakatau Project Manager	x	x	x	x (CDF/CSV)	-	-	-
Krakatau Professional **Krakatau Lite**	x	x	x	x (CDF/CSV)	-	-	-
Resource Standard Metrics – RSM / RSM Wizard (GUI)	x	-	x	x (text, CSV)	-	-	-
TychoMetrics	x	x	x	x (CSV, MetaFile, BMP, JPG, PNG)	- (x) [not confirmed]	- (x) [not confirmed]	-
Appraisal Wizard	x	x	x	x (text, CSV, Excel, rtf, pdf, bmp, jpg, wmf, emf, gif, wb1, wk2, dif, slk)	-	-	-
Metrics4C, Metrics4FORTRAN, Metrics4Pascal, Metrics4Project	x	x	-	x (text[console output], gif)	-	-	-
ASSESS for Assembler, COBOL, Java	x	-	x	x (with external Plugins: CSV,…)	x (MSAccess)	x	-
ExperiencePro	x	x	x	x (txt, xls)	-	-	-
CodeReports	x	x	x	x (xls, Crystal Reports, …)	x (ODBC, JDBC)	-	-
CodeCheck	x	-	*(customizeable)	x (txt) *customizeable	-	* (customizeable)	-
MPP (Kuhrau)	x	-	-	x (txt,xls)	-	-	-
COSMOS	x	-	-	x (txt)	-	-	-
LDRA Testbed	x	x	x	x (txt)	-	-	-
JUNIT	x	x	x	x (txt)	-	x	-
JDepend	x	x	x (with Ant)	x (txt)	-	x	-

Appendix D: Process Modeling with BPMN

To present an overview about the Business Process Modelling Notation the following short description has been taken from selectbs.com (http://www.selectbs.com/adt/analysis-and-design/what-is-business-process-modeling-notation-bpmn)

The Business Process Modeling Notation (BPMN) is a standardized graphical notation for drawing business processes in a workflow.

The Business Process Management Initiative has developed a standard Business Process Modeling Notation (BPMN) [OMG06]. The primary goal of BPMN is to provide a notation that is readily understandable by all business users, from the business analysts that create the initial drafts of the processes, to the technical developers responsible for implementing the technology that will perform those processes, and finally, to the business people who will manage and monitor those processes. Thus, BPMN creates a standardized bridge for the gap between the business process design and process implementation.

This specification defines the notation and semantics of a Business Process Diagram (BPD) and represents the amalgamation of best practices within the business modeling community. The intent of BPMN is to standardize a business process modeling notation in the face of many different modeling notations and viewpoints. In doing so, BPMN will provide a simple means of communicating process information to other business users, process implementers, customers, and suppliers.

There are different levels of process modeling [OMG06]:

- Process Maps – simple flow charts of the activities
- Process Descriptions – flow charts extended with additional information, but not enough to fully define actual performance
- Process Models – flow charts extended with enough information so that the process can be analyzed, simulated, and/or executed
- BPMN supports each of these levels

A goal for the development of BPMN is that the notation be simple and adoptable by business analysts. The modeling BPMN is made by simple diagrams with a small set of graphical elements. It should make it easy for business user as well as developers to understand the flow and the process. The four basic categories of elements are:

1. Flow Objects
    a. Events
    b. Activities

  c. Gateways

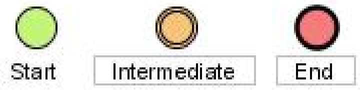
2. Connecting Objects
   a. Sequence flow
   b. Message Flow
   c. Association
3. Swim lanes
   a. Pool
   b. Lane
4. Artifacts
   a. Data Objects
   b. Group
   c. Annotation

These four categories of elements give us the opportunity to make a simple diagram (BPD). It is also allowed in BPD to make your own type of a Flow Object or an Artifact to make the diagram more understandable.

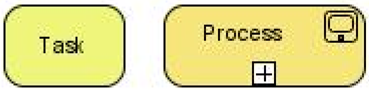
Flow objects

Flow Objects consist of only three core elements. The three Flow Objects are:

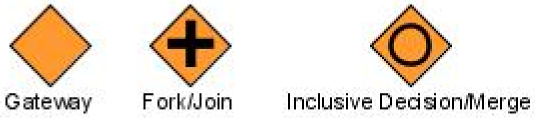
- Event: An Event is represented with a circle and is something that happens. It could be Start, Intermediate or End. This element is a trigger or a result.



- Activity: An Activity is represented with a rounded-corner rectangle and shows us the kind of work which must be done. It could be a task or a sub-process. A sub-process also has a plus sign in the bottom line of the rectangle.



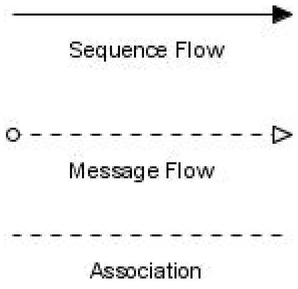
- Gateway: A Gateway is represented with a diamond shape and will determine different decisions. It will also determine forking, merging and joining of paths [OMG06].



Connecting objects

The Flow Objects are connected to each other with Connecting Objects. There are three different Connecting Objects [OMG06]:

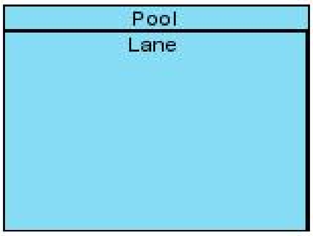
- Sequence Flow: A Sequence Flow is represented with a solid line and arrowhead and shows in which order the activities will be performed. A diagonal slash across the line close to the origin indicates a default choice of a decision.
- Message Flow: A Message Flow is represented with a dashed line and an open arrowhead. It tells us what messages flow between two process participants.
- Association: An Association is represented with a dotted line and a line arrowhead. It is used to associate an Artifact, data or text to a Flow Object.



Swim lanes

A Swim lane is a visual mechanism of organizing different activities into categories of the same functionality. There are two different swim lanes, and they are:

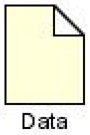
- Pool: A Pool is represented with a big rectangle which contains many Flow Objects, Connecting Objects and Artifacts.
- Lane: A Lane is represented as a sub-part of the pool. The lanes are used to organize the Flow Objects, Connecting Objects and Artifacts more precisely.



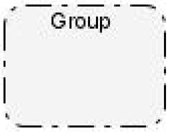
Artifacts

Artifacts allow developers to bring some more information into the model/diagram. In this way the model/diagram becomes more readable. There are three pre-defined Artifacts and they are [OMG06]:

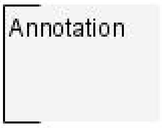
- Data Objects: Data Objects are used to show the reader which data is required or produced in an activity.



- Group: A Group is represented with a rounded-corner rectangle and dashed lines. The Group is used to group different activities but does not affect the flow in the diagram.



- Annotation: An Annotation is used to give the reader of the model/diagram an understandable impression.



Appendix E: Detailed Description of selected Artifacts

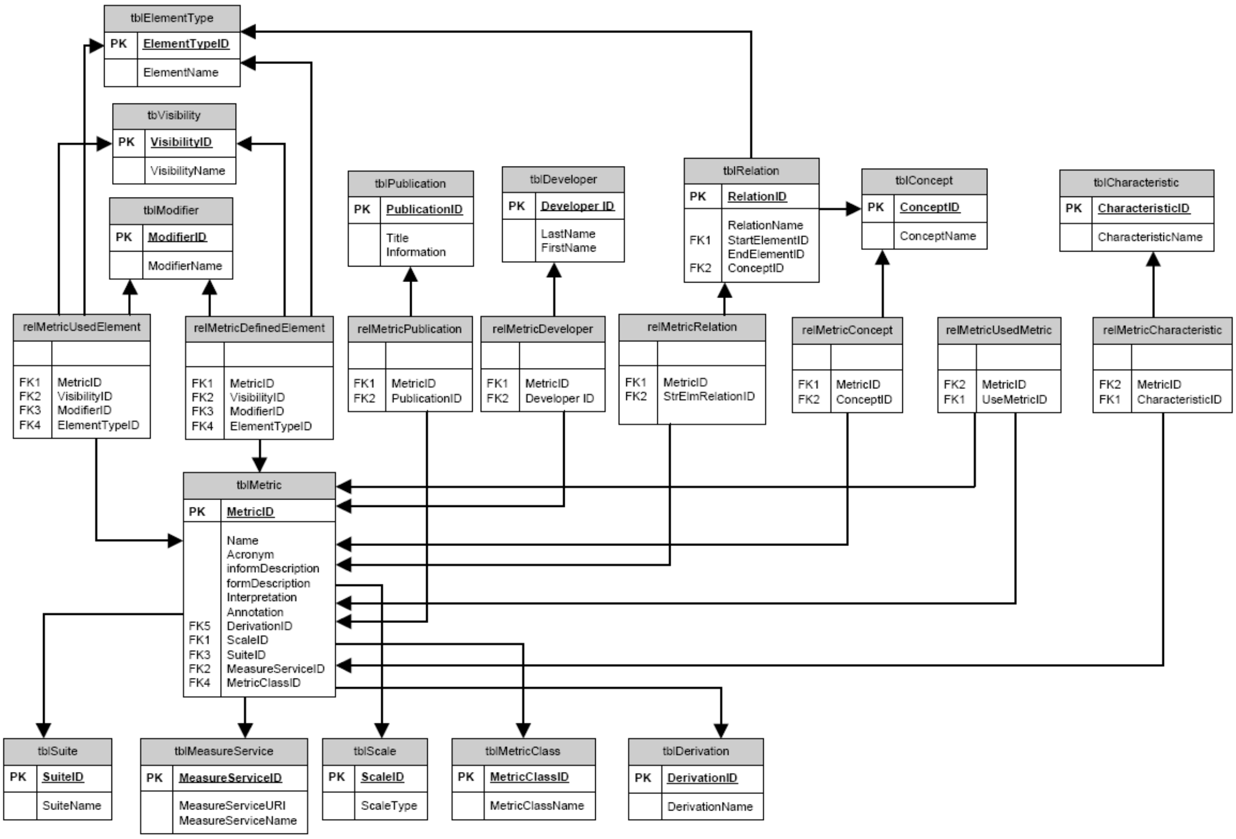


Figure 118: Schema of the OOMO database [Weise06]

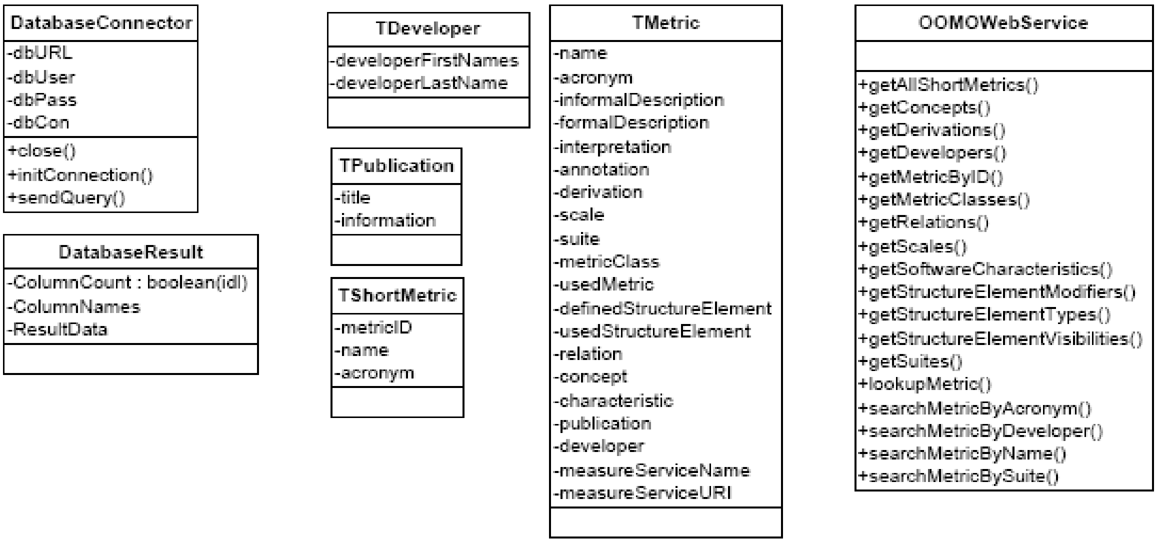
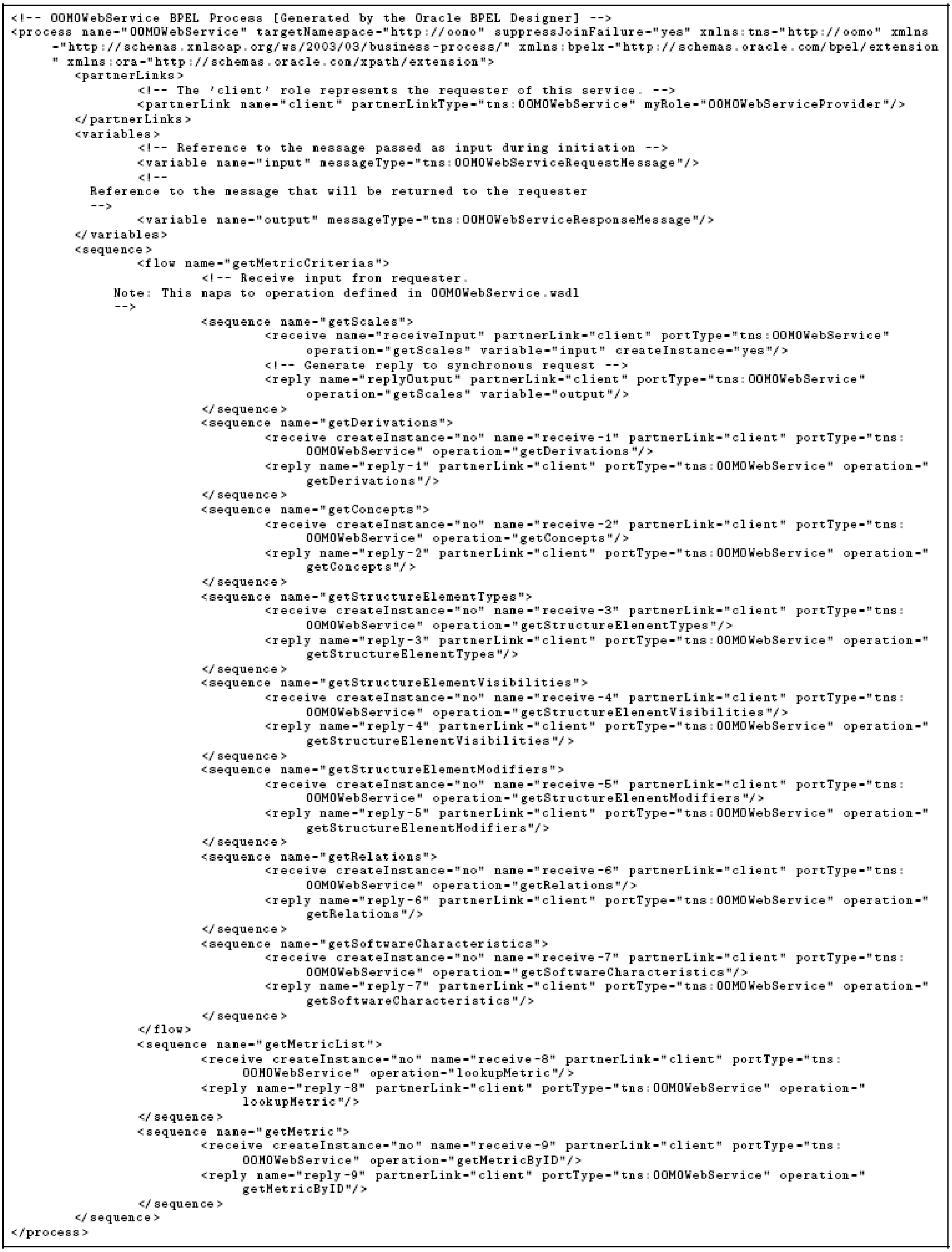


Figure 119: OOMO class description



```
<!-- OOMOWebService BPEL Process [Generated by the Oracle BPEL Designer] -->
<process name="OOMOWebService" targetNamespace="http://oomo" suppressJoinFailure="yes" xmlns:tns="http://oomo" xmlns
      ="http://schemas.xmlsoap.org/ws/2003/03/business-process/" xmlns:bpelx="http://schemas.oracle.com/bpel/extension
      " xmlns:ora="http://schemas.oracle.com/xpath/extension">
        <partnerLinks>
                <!-- The 'client' role represents the requester of this service. -->
                <partnerLink name="client" partnerLinkType="tns:OOMOWebService" myRole="OOMOWebServiceProvider"/>
        </partnerLinks>
        <variables>
                <!-- Reference to the message passed as input during initiation -->
                <variable name="input" messageType="tns:OOMOWebServiceRequestMessage"/>
                <!--
          Reference to the message that will be returned to the requester
          -->
                <variable name="output" messageType="tns:OOMOWebServiceResponseMessage"/>
        </variables>
        <sequence>
                <flow name="getMetricCriterias">
                        <!-- Receive input from requester.
            Note: This maps to operation defined in OOMOWebService.wsdl
            -->
                        <sequence name="getScales">
                                <receive name="receiveInput" partnerLink="client" portType="tns:OOMOWebService"
                                     operation="getScales" variable="input" createInstance="yes"/>
                                <!-- Generate reply to synchronous request -->
                                <reply name="replyOutput" partnerLink="client" portType="tns:OOMOWebService"
                                     operation="getScales" variable="output"/>
                        </sequence>
                        <sequence name="getDerivations">
                                <receive createInstance="no" name="receive-1" partnerLink="client" portType="tns:
                                     OOMOWebService" operation="getDerivations"/>
                                <reply name="reply-1" partnerLink="client" portType="tns:OOMOWebService" operation="
                                     getDerivations"/>
                        </sequence>
                        <sequence name="getConcepts">
                                <receive createInstance="no" name="receive-2" partnerLink="client" portType="tns:
                                     OOMOWebService" operation="getConcepts"/>
                                <reply name="reply-2" partnerLink="client" portType="tns:OOMOWebService" operation="
                                     getConcepts"/>
                        </sequence>
                        <sequence name="getStructureElementTypes">
                                <receive createInstance="no" name="receive-3" partnerLink="client" portType="tns:
                                     OOMOWebService" operation="getStructureElementTypes"/>
                                <reply name="reply-3" partnerLink="client" portType="tns:OOMOWebService" operation="
                                     getStructureElementTypes"/>
                        </sequence>
                        <sequence name="getStructureElementVisibilities">
                                <receive createInstance="no" name="receive-4" partnerLink="client" portType="tns:
                                     OOMOWebService" operation="getStructureElementVisibilities"/>
                                <reply name="reply-4" partnerLink="client" portType="tns:OOMOWebService" operation="
                                     getStructureElementVisibilities"/>
                        </sequence>
                        <sequence name="getStructureElementModifiers">
                                <receive createInstance="no" name="receive-5" partnerLink="client" portType="tns:
                                     OOMOWebService" operation="getStructureElementModifiers"/>
                                <reply name="reply-5" partnerLink="client" portType="tns:OOMOWebService" operation="
                                     getStructureElementModifiers"/>
                        </sequence>
                        <sequence name="getRelations">
                                <receive createInstance="no" name="receive-6" partnerLink="client" portType="tns:
                                     OOMOWebService" operation="getRelations"/>
                                <reply name="reply-6" partnerLink="client" portType="tns:OOMOWebService" operation="
                                     getRelations"/>
                        </sequence>
                        <sequence name="getSoftwareCharacteristics">
                                <receive createInstance="no" name="receive-7" partnerLink="client" portType="tns:
                                     OOMOWebService" operation="getSoftwareCharacteristics"/>
                                <reply name="reply-7" partnerLink="client" portType="tns:OOMOWebService" operation="
                                     getSoftwareCharacteristics"/>
                        </sequence>
                </flow>
                <sequence name="getMetricList">
                        <receive createInstance="no" name="receive-8" partnerLink="client" portType="tns:
                             OOMOWebService" operation="lookupMetric"/>
                        <reply name="reply-8" partnerLink="client" portType="tns:OOMOWebService" operation="
                             lookupMetric"/>
                </sequence>
                <sequence name="getMetric">
                        <receive createInstance="no" name="receive-9" partnerLink="client" portType="tns:
                             OOMOWebService" operation="getMetricByID"/>
                        <reply name="reply-9" partnerLink="client" portType="tns:OOMOWebService" operation="
                             getMetricByID"/>
                </sequence>
        </sequence>
</process>
```

Figure 120: BPEL structure of the OOMO metrics identification process