Nr.: FIN-013-2008

Formal Modelling of Software Measurement Levels of Paradigm-Based Approaches

R. Dumke, M. Kunz, A. Farooq, K. Georgieva, H. Hegewald

Arbeitsgruppe Softwaretechnik

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Technical Report

Nr.: FIN-013-2008

Formal Modelling of Software Measurement Levels of Paradigm-Based Approaches

R. Dumke, M. Kunz, A. Farooq, K. Georgieva, H. Hegewald

Arbeitsgruppe Softwaretechnik

Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg

Impressum (§ 10 MDStV):

Herausgeber:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Der Dekan

Verantwortlich für diese Ausgabe:
Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik
Reiner Dumke
Postfach 4120
39016 Magdeburg
E-Mail: dumke@ovgu.de

http://www.cs.uni-magdeburg.de/Preprints.html

Auflage: 101

Redaktionsschluss: 31.10.2008

Herstellung: Dezernat Allgemeine Angelegenheiten,
Sachgebiet Reproduktion

Bezug: Universitätsbibliothek/Hochschulschriften- und
Tauschstelle

Formal Modelling of Software Measurement Levels of Paradigm-Based Approaches

Reiner Dumke, Martin Kunz, Ayaz Farooq, Konstantina Georgieva, Heike Hegewald

SML@b, University of Magdeburg, Germany, http://www.smlab.de

Contents

1 Introduction

Formal descriptions of software measurement can be found in the following kinds of exemplary motivations,

- *Understanding* the essential components, operations, methodologies and empirical background of this special kind of measurement
- *Clarification* of the different scale types of metrics or measures considering the different software process areas as product processes and resources
- *Foundation* as a theoretical basis for classification, structuring and formal proving of the software measurement paradigms.

The following figure shows some kinds of formal approaches that could be found in the literature (see [Dumke 2005c] for detailed descriptions).

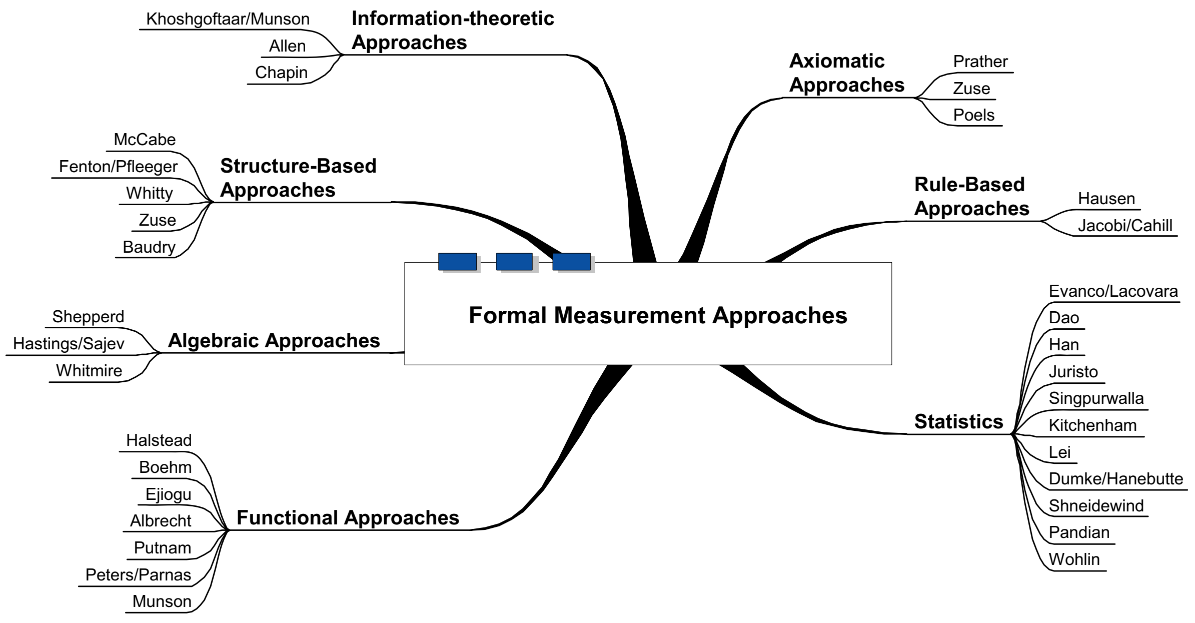


Figure 1: Formal approaches of software measurement

Considering the measurement systems aspects we define a *software measurement system* in a declarative manner as following ([Dumke 2005c], [Skyttner 2005]):

$$\boldsymbol{MS} = (M_{MS}, R_{MS}) = (\{\boldsymbol{G}, \boldsymbol{A}, \boldsymbol{M}, \boldsymbol{Q}, \boldsymbol{V}, \boldsymbol{U}, \boldsymbol{E}, \boldsymbol{T}, \boldsymbol{P}\}, R_{MS}) \qquad \textbf{(1.1)}$$

where $\boldsymbol{G}$ is the set of the measurement goals, $\boldsymbol{A}$ the set of measured artefacts or measurement objects, $\boldsymbol{M}$ the set of measurement methods, objects or entities, $\boldsymbol{Q}$ the set of measurement quantities, $\boldsymbol{V}$ the set of measurement values (especially we could have the situation $\boldsymbol{Q} = \boldsymbol{V}$), $\boldsymbol{U}$ the set of measurement units, $\boldsymbol{E}$ the set of measurement-based experience, $\boldsymbol{T}$ the set of measurement CASE tools (respectively CAME tools), and $\boldsymbol{P}$ the set of the measurement personnel. R_{MS} defines all meaningful relations between the elements of M_{MS}. Note that our description involves the principles of goal question metric (GQM), SPICE and CMMI measurement intentions and fulfils the basic characteristics of the ISO 15939 software measurement standard shown in the following figure [ISO 15939].

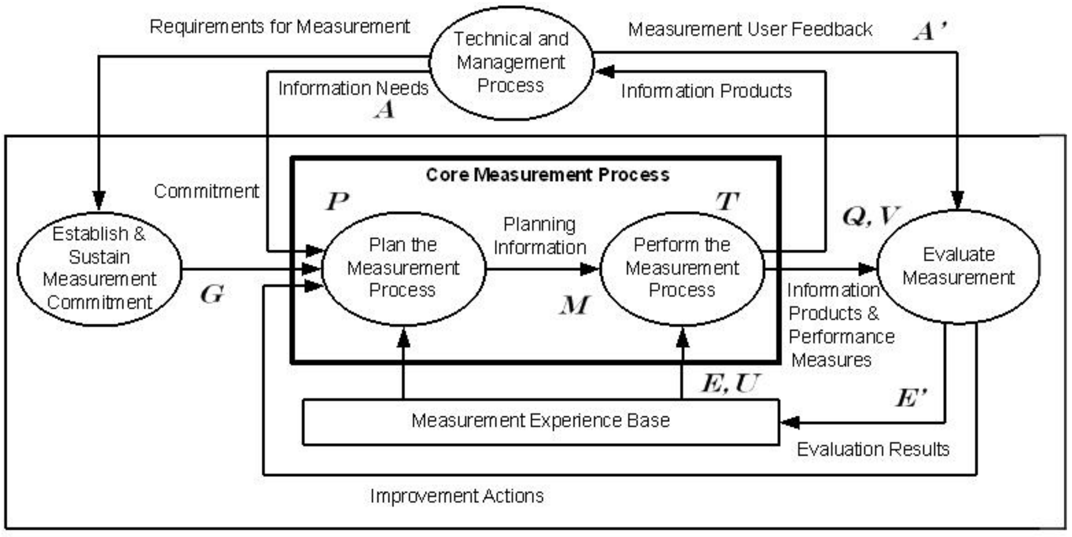

Requirements for Measurement
Technical and Management Process
Measurement User Feedback
A'
Information Needs
A
Information Products
Core Measurement Process
Commitment
P
T
Q, V
Establish & Sustain Measurement Commitment
G
Plan the Measurement Process
Planning Information
M
Perform the Measurement Process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

Figure 2: The ISO 15939 software measurement standard

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

$$MP: \quad (G \times A \times M)_{T,P} \rightarrow (Q \times E)_{T,P} \rightarrow (V \times U)_{T,P} \rightarrow E' \times A' \qquad \textbf{(1.2)}$$

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.

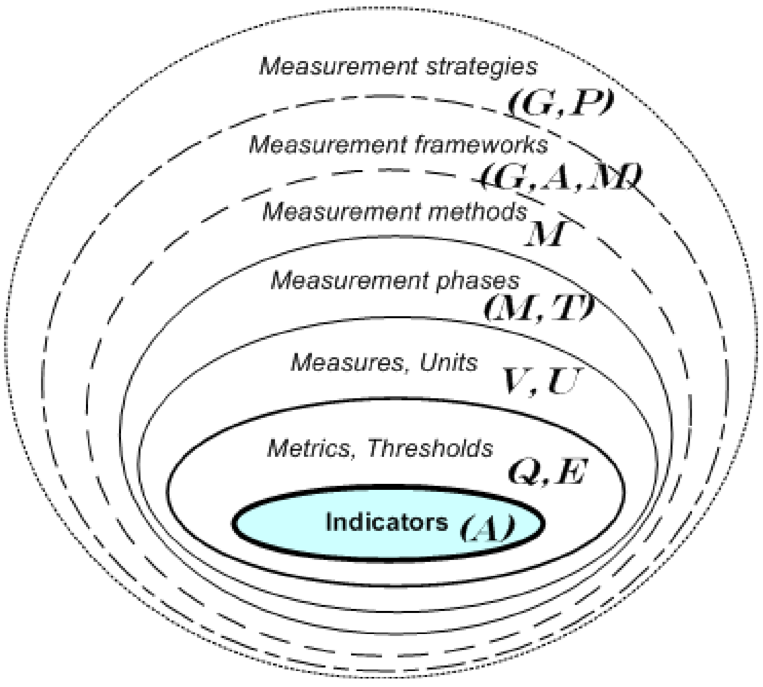


Figure 3: The general layer model of software measurement

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

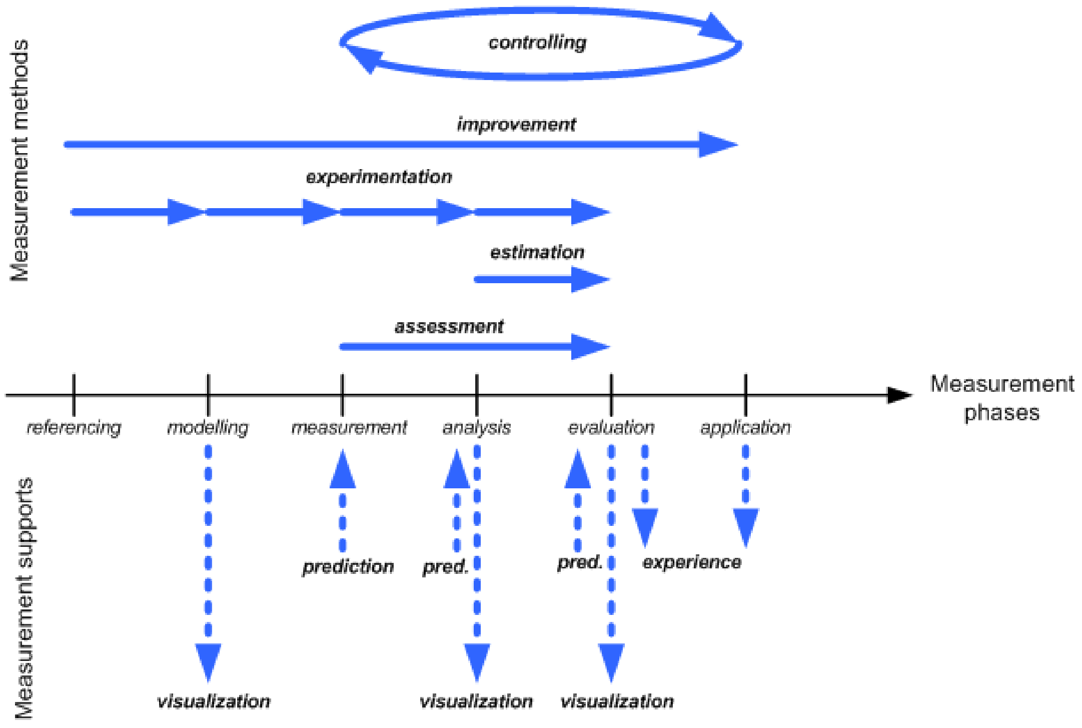


Figure 4: Software measurement phases and methods

Finally the kernel consideration what software measurement is could be characterized as homomorphous relationship verbally described in the following figure.

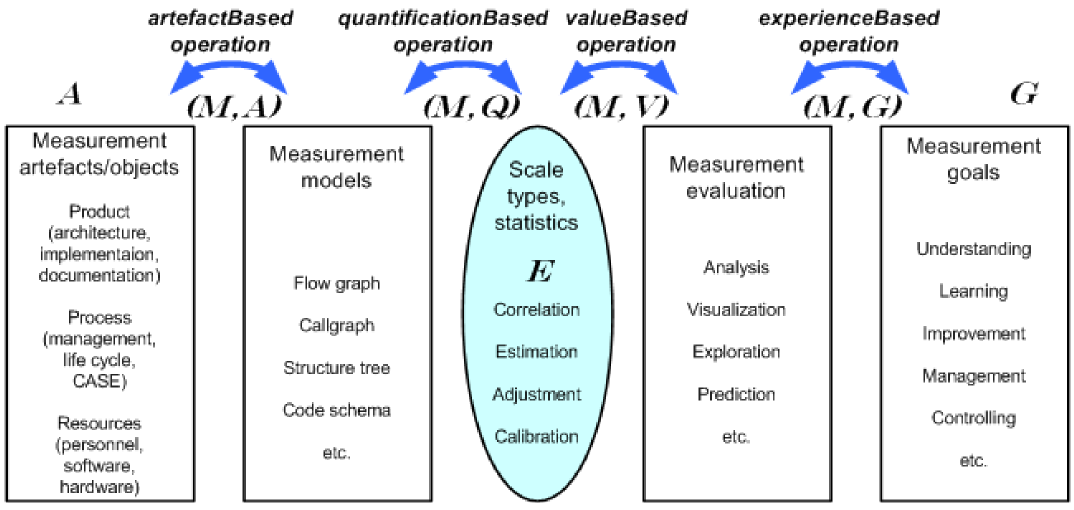


Figure 5: The homomorphous relationship of software measurement

Based on our software measurement experiences we can derive the following refinement[1] on the process description above ([Braungarten 2007], [Dumke 1999], [Dumke 2005a], [Dumke 2006a], [Ebert 2007], [Rud 2006], [Schmietendorf 2002]).

[1] This refinement does not fulfil the principle of completeness.

1.1 Measurement Ingredients

The tuple of ($\boldsymbol{G} \times \boldsymbol{A} \times \boldsymbol{M}$) describes the input and basis for any software measurement. The detailed characteristics of these three sets are[2] :

***G*: Intention:** We will consider in our approach the goals 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.*

***A*: Domain:** The considered measurement artefacts 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*)).

Origin: Note that we could consider a pendant or analogical artefact of measurement that led us to the kinds of measurement as *analogical conclusion*. Analogy can be defined as *tuning* (where we use a pendant in the same class of software systems) and as *adaptation* (where we use another pendant of artefact). This kind of description is motivated in the following consideration below.

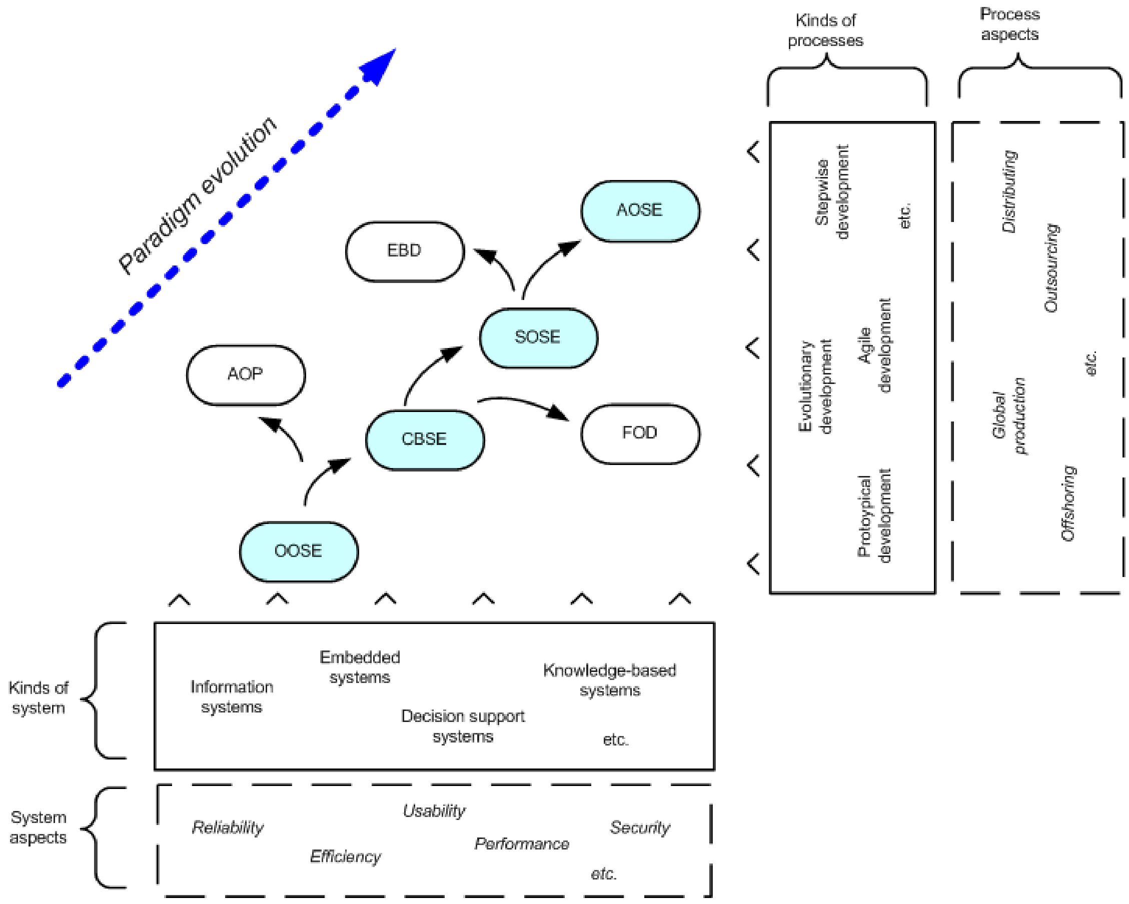


Figure 6: The complexity of software as measurement artefacts

[2] Note that we will define *two* characteristics for every set as two types of classification.

The complexity of the measured artefact could be explained as following: 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).

On the other hand, 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. Figure 6 shows the relationships between these characteristics in a simplified manner.

M: ***Method:*** The chosen measurement methods should be classified here as *experiment/case study, assessment, improvement* and *controlling.* That means that measurement should contain the partial phases as *referencing, modelling, 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.*

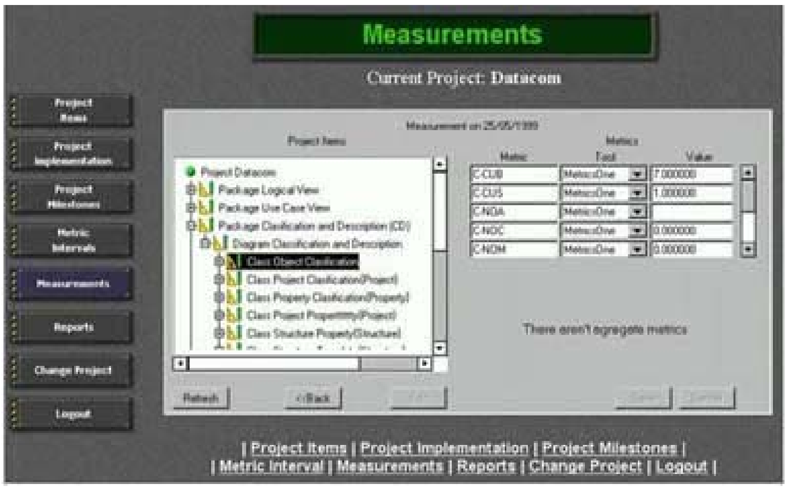
Sort: Furthermore, depending on the measured artefact(s) that is involved in the measurement we will distinguish between *no measurement* (no artefact), *aspect-oriented measurement* (considering some aspects of product or process or resources), *capability-oriented measurement* (considering the whole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

1.2 Measurement Output

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

Q: ***Value:*** This set of metrics values/numbers characterises 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.*


Measurements
Current Project: Datacom
Project Items
Project Implementation
Project Milestones
Metric Intervals
Measurements
Reports
Change Project
Logout
Measurement on 25/05/1999
Project Items
Project Datacom
Package Logical View
Package Use Case View
Package Clasification and Description (CD)
Diagram Clasification and Description
Class Object Clasification
Class Project Clasification(Project)
Class Property Clasification(Property)
Class Project Property(Project)
Class Structure Property(Structure)
Refresh
<Back
There aren't agregate metrics
| Project Items | Project Implementation | Project Milestones |
| Metric Interval | Measurements | Reports | Change Project | Logout |

Metric	Tool	Value
C-CUB	MetricsOne	7.000000
C-CUS	MetricsOne	1.000000
C-NOA	MetricsOne	
C-NOC	MetricsOne	0.000000
C-NOM	MetricsOne	0.000000

Figure 7: UML based metrics data base example

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

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

Typical kinds of measurement repositories are *metrics databases*. The example of such a data base was the one used in T-Systems for UML based product measurement [Ebert 2007] as given in figure 7.

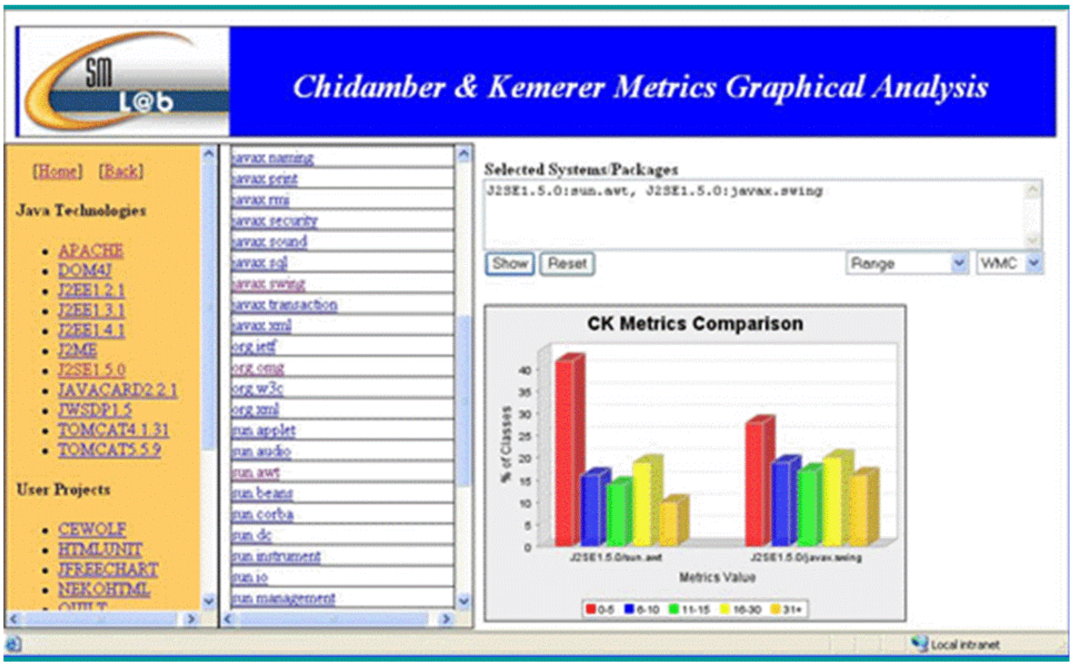


Figure 8: The java measurement service OOMJ

An excellent variety of measurement quantities is stored in the Java measurement repository as OOMJ implemented by Farooq [Farooq 2005] as shown in the figure 8.

1.3 Measurement Results

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

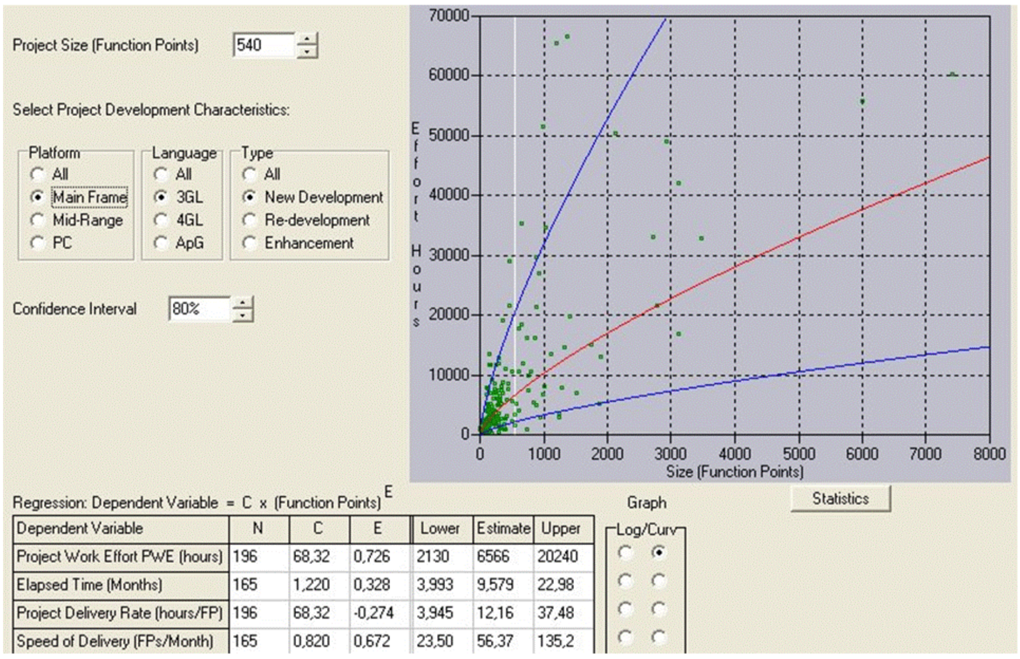
V: ***Measure:*** This set of metrics values characterises a *quantitative measurement* and is given an *interval scale* or *ratio scale.*

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, *kilo delivered lines of code* (KDSI), *cyclomatic complexity* of McCabe etc.

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

The ISBSG (International Software Benchmarking Standards Group) is an international community that summarizes the project management data from IT companies worldwide. The following figure shows an example evaluation of some of these measurement values [ISBSG 2003].



Project Size (Function Points): 540

Select Project Development Characteristics:

Platform: All [✓ Main Frame] Mid-Range PC

Language: All [✓ 3GL] 4GL ApG

Type: All [✓ New Development] Re-development Enhancement

Confidence Interval: 80%

Regression: Dependent Variable = C x (Function Points)^E

Dependent Variable	N	C	E	Lower	Estimate	Upper
Project Work Effort PWE (hours)	196	68,32	0,726	2130	6566	20240
Elapsed Time (Months)	165	1,220	0,328	3,993	9,579	22,98
Project Delivery Rate (hours/FP)	196	68,32	-0,274	3,945	12,16	37,48
Speed of Delivery (FPs/Month)	165	0,820	0,672	23,50	56,37	135,2

Graph – Log/Curv: Project Work Effort PWE (hours) [✓ Curv]

Figure 9: The ISBSG project measurement repository

1.4 Measurement Resources

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

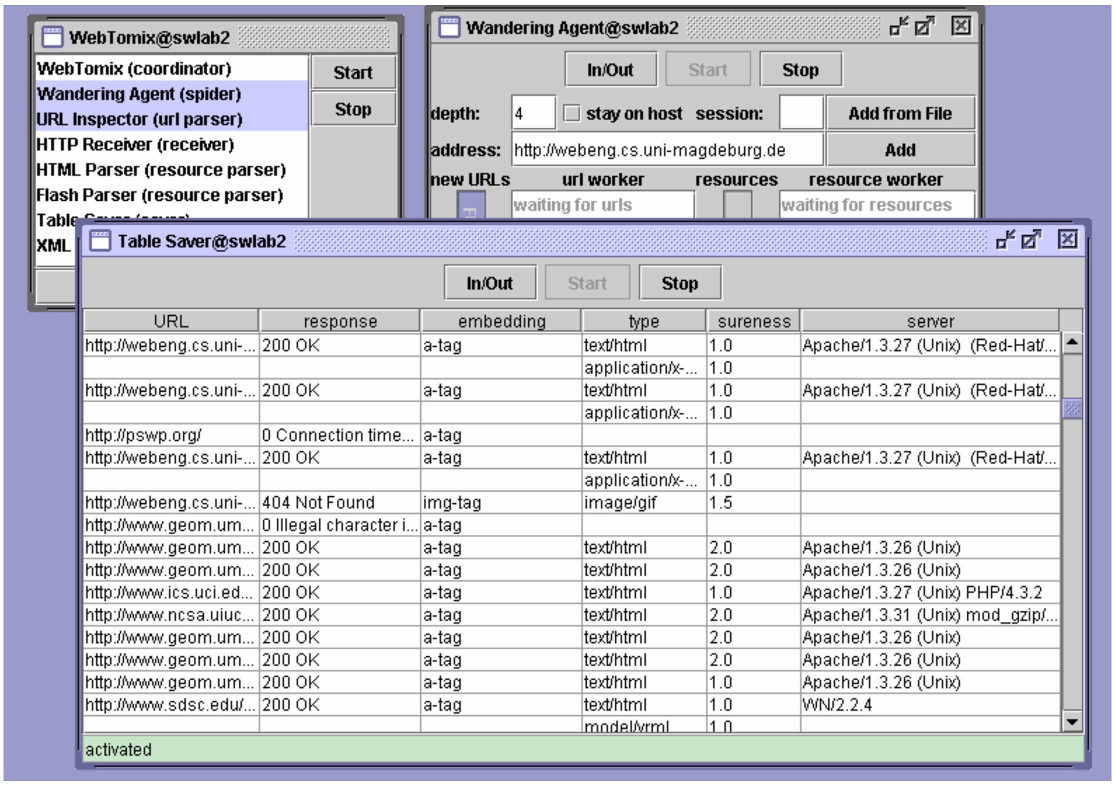
T: Level: The kind of tool and the tool support should be classified as *manual* (without any tools), *semi-automatic* and *automatic.*

Support: On the other hand 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 could be 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).

An example of measurement tool using the tomograph methodology that differs the phase of measurement and evaluation is shown in the following figure for Web measurement [Dumke 2003].



WebTomix@swlab2

WebTomix (coordinator)
Wandering Agent (spider)
URL Inspector (url parser)
HTTP Receiver (receiver)
HTML Parser (resource parser)
Flash Parser (resource parser)
Table
XML

Start | Stop

Wandering Agent@swlab2

In/Out | Start | Stop

depth: 4 | stay on host | session: | Add from File

address: http://webeng.cs.uni-magdeburg.de | Add

new URLs | url worker | resources | resource worker

waiting for urls | waiting for resources

Table Saver@swlab2

In/Out | Start | Stop

URL	response	embedding	type	sureness	server
http://webeng.cs.uni-...	200 OK	a-tag	text/html	1.0	Apache/1.3.27 (Unix) (Red-Hat/...
			application/x-...	1.0	
http://webeng.cs.uni-...	200 OK	a-tag	text/html	1.0	Apache/1.3.27 (Unix) (Red-Hat/...
			application/x-...	1.0	
http://pswp.org/	0 Connection time...	a-tag			
http://webeng.cs.uni-...	200 OK	a-tag	text/html	1.0	Apache/1.3.27 (Unix) (Red-Hat/...
			application/x-...	1.0	
http://webeng.cs.uni-...	404 Not Found	img-tag	image/gif	1.5	
http://www.geom.um...	0 Illegal character i...	a-tag			
http://www.geom.um...	200 OK	a-tag	text/html	2.0	Apache/1.3.26 (Unix)
http://www.geom.um...	200 OK	a-tag	text/html	2.0	Apache/1.3.26 (Unix)
http://www.ics.uci.ed...	200 OK	a-tag	text/html	1.0	Apache/1.3.27 (Unix) PHP/4.3.2
http://www.ncsa.uiuc...	200 OK	a-tag	text/html	2.0	Apache/1.3.31 (Unix) mod_gzip/...
http://www.geom.um...	200 OK	a-tag	text/html	2.0	Apache/1.3.26 (Unix)
http://www.geom.um...	200 OK	a-tag	text/html	2.0	Apache/1.3.26 (Unix)
http://www.geom.um...	200 OK	a-tag	text/html	1.0	Apache/1.3.26 (Unix)
http://www.sdsc.edu/...	200 OK	a-tag	text/html	1.0	WN/2.2.4
			model/vrml	1.0	

activated

Figure 10: The Web tomograph tool layout for Web measurement

Furthermore, supporting the agile software development, a helpful solution consists in the metrics-based evaluation of the stepwise implemented results shown in the following figure.

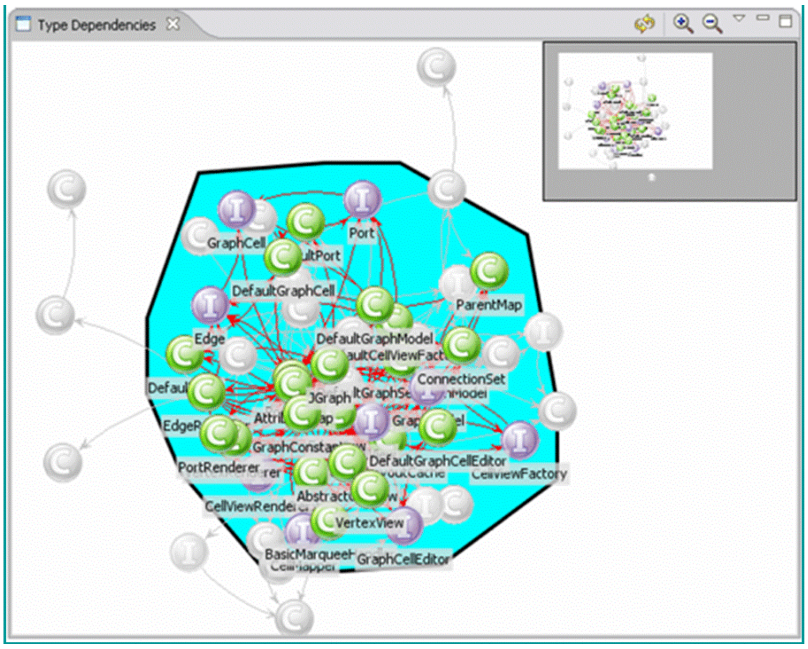


Figure 11: The tool-based metrics evaluation of agile software development

1.5 Measurement Repercussions

Finally, the software measurement could/should lead to extensions of the experience and to improvements of the measures artefacts explained in the tuple ($\mathcal{E}' \times \mathcal{A}'$). Typical properties are:

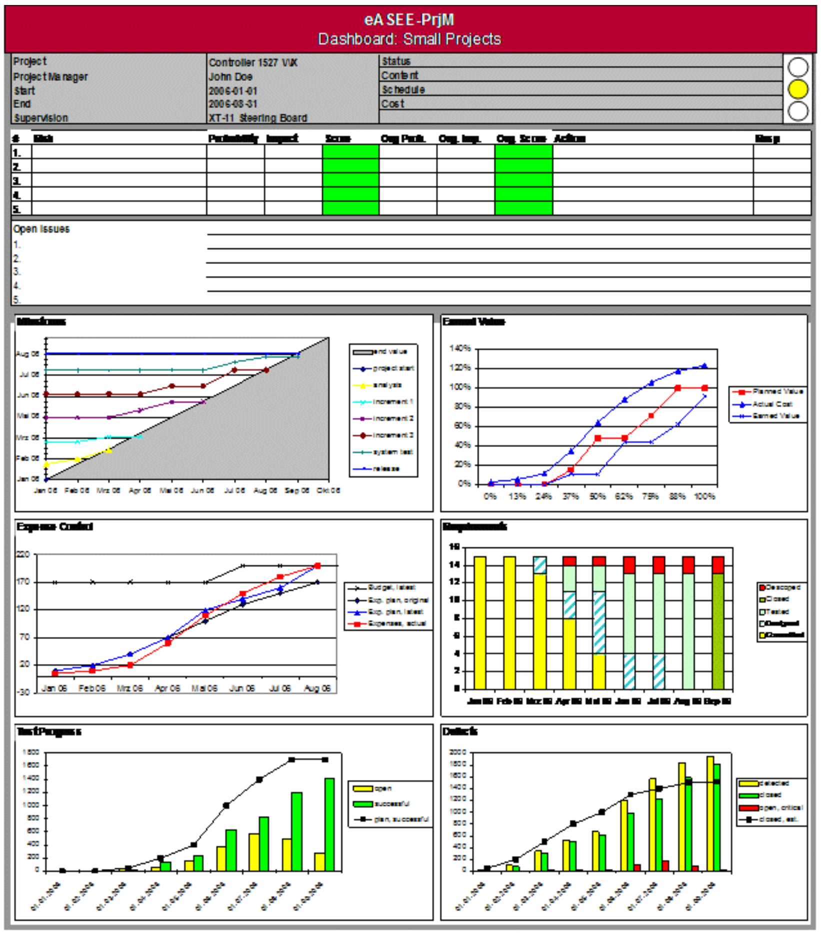
***E'*: Form:** The obtained experiences are also given as *analogies, axioms, correlations, intuitions, laws, trends, lemmas, formulas, principles, conjectures, hypothesises* and *rules of thumb.*

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 artefacts leads to such a marked set $\mathcal{A}$**.**

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

In the IT practice the summarizing of measurement results as cockpits or dashboards is helpful in order to achieve a holistic view.


eA SEE-PrjM
Dashboard: Small Projects

Project	Controller 1527 V/X	Status	○
Project Manager	John Doe	Content	●
Start	2006-01-01	Schedule	○
End	2006-08-31	Cost	
Supervision	XT-11 Steering Board		

#	Risk	Probability	Impact	Score	Opp Prob.	Opp. Imp.	Opp. Score	Action	Resp
1.				🟩			🟩		
2.				🟩			🟩		
3.				🟩			🟩		
4.				🟩			🟩		
5.				🟩			🟩		

Open Issues
1.
2.
3.
4.
5.

Figure 12: Example of a software measurement dashboard [Ebert 2007]

1.6 General Characterization of Software Measurement Process

The measurement process $\mathcal{MP}$ itself should be characterized by the level of covered/measured artifacts (as ***approach***) and by the kind of IT relationship (as ***solution***). Hence, we could define the essential measurement process characteristics in the following formal manner [Dumke 2007]:

$$\mathcal{MP}^{\,approach}_{\,solution}: \tag{1.3}$$

$$(G^{intention}_{viewpoint} \times A^{domain}_{origin} \times M^{method}_{sort})_{T^{level}_{support}, P^{kind}_{area}} \rightarrow (Q^{value}_{structure} \times E^{form}_{contents})_{T^{level}_{support}, P^{kind}_{area}}$$

$$\rightarrow (V^{measure}_{aggregation} \times U^{type}_{standard})_{T^{level}_{support}, P^{kind}_{area}} \rightarrow E^{form}_{extension} \times A^{domain}_{changing}$$

The classification of the measurement process $\mathcal{MP}$ itself is based on the measured artefact. The measurement of aspects (aspects product or processes or resources) leads us 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 artefacts (product and process and resources) we will call this as a *whole measurement*. These characteristics build the "approach" attribute of measurement 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*.

Finally, further information about examples of software measurement methods and processes can be found in the Software Measurement Laboratory at the University of Magdeburg (SML@b) at http://www.smlab.de.

Software Measurement Laboratory SML@b

COMMUNITY
- Conferences/Workshops
- Measurement Associations

PUBLICATIONS
- Bibliographies
- SMLab's Publications
- Standards

METRICS TOOLS
- Tool Overviews
- Web-based Measurement

EXPERIENCE
- Motivation
- Experiments

RESEARCH PROJECTS

TEACHING

GI Fachgruppe Metriken (2.1.10)

Otto-von-Guericke-University

PUBLICATIONS

Software Measurement

"Experience-Based Software Measurement and Evaluation Considering Paradigm Evolution" (Metrikon paper)

"Towards a Service-Oriented Measurement Infrastructure" (SMEF paper)

"Challenges in Editing Books on Software Measurement" (IWSM/Mensura 2007 Keynote)

NEWS

November 2008: IWSM/Metrikon/Mensura and BSOA Conferences

October 2008: Japanese Scientists at the SML@b

Prof. Dumke(SML@b), Dr. Mitani (SE Center, Tokyo), Prof. Matsumoto (NAIST, Ikoma), Prof. Takahashi (SE Center, Tokyo), DI Kunz(SML@b)

RESEARCH PROJECTS

The Java Measurement Service

As integral part of the master thesis of A. Farooq a web-based Java Measurement Portal has been developed supporting OO Metrics Evaluation about different Java versions and techologies...

READ MORE >>

Web Service Trust Center

The Web Service Trust Center integrates tools for measurement and analysis of performance, availability and other quality of service (QoS) aspects of web service infrastructures...

READ MORE >>

Figure 13: The SML@b portal

2 Software Measurement Process Levels

2.1 Basics of Scalability

In this section we give a first graduation of the software measurement characteristics introduced in the section 1. 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* [Zelkowitz 2007].
2. A consideration of the experiment levels by Kitchenham leads to (also decreasing): *industrial case studies, quasi experiment*, and *formal experiment* [Kitchenham 2007].
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* [Sneed 2005].

We will use these experiences and some of the results from our industrial projects at Alcatel, Siemens, Bosch and German Telekom ([Braungarten 2005], [Dumke 2007], [Ebert 2007], [Richter 2005], [Schmietendorf 2007] and [Wille 2005]) 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 [Bourque 2007], [Braungarten 2007], [Farooq 2005], [Laird 2006], [Pandian 2004], [Schmietendorf 2007] and [Sneed 2005]). Our first ordinal classifications of the measurement process components in an increasing manner are the following

$$\begin{aligned}
\mathbf{G:}\quad & \mathit{intention} \in \{\textbf{understanding, evaluation, improving, managing}\} && (2.1)\\
& \mathit{viewpoint} \in \{\textbf{internal\_goals, external\_goals, goals\_in\_use}\}
\end{aligned}$$

$$\begin{aligned}
\mathbf{A:}\quad & \mathit{domain} \in \{(\textbf{product\_aspects} \vee \textbf{process\_aspects} \vee \textbf{resources\_aspects}), && (2.2)\\
& \qquad (\textbf{product} \vee \textbf{process} \vee \textbf{resources}), (\textbf{product} \wedge \textbf{process} \wedge \textbf{resources})\}\\
& \mathit{origin} \in \{\textbf{other\_pendant, pendant\_in\_same\_domain, original}\}
\end{aligned}$$

$$\begin{aligned}
\mathbf{M:}\quad & \mathit{method} \in \{\textbf{experimen/case study, assessment, improvement, controlling}\} && (2.3)\\
& \mathit{sort} \in \{\textbf{analogical\_conclusion, estimation, simulation, measurement}\}
\end{aligned}$$

$$\begin{aligned}
\mathbf{T:}\quad & \mathit{level} \in \{\textbf{manual, semi-automatic, automatic}\} && (2.4)\\
& \mathit{support} \in \{\textbf{one\_measurement\_phase, some\_measurement\_phases, whole\_measurement}\}
\end{aligned}$$

$$\begin{aligned}
\mathbf{P:}\quad & \mathit{kind} \in \{\textbf{manager, researcher, practitioner}\} && (2.5)\\
& \mathit{area} \in \{\textbf{measurement\_expert\_staff, measurement\_application\_staff}\}
\end{aligned}$$

$$\begin{aligned}
\mathbf{Q:}\quad & \mathit{value} \in \{\textbf{identifier/nomination, ordinal\_scale}\} && (2.6)\\
& \mathit{structure} \in \{\textbf{single\_value}, (\textbf{normalization} \vee \textbf{transformation}), \textbf{aggregation}\}
\end{aligned}$$

$$\begin{aligned}
\mathbf{E:}\quad & \mathit{form} \in \{(\textbf{intuition} \vee \textbf{law} \vee \textbf{trend} \vee \textbf{principle}), \textbf{analogy}, (\textbf{criteria} \vee \textbf{rules\_of\_thumb}),\\
& \qquad (\textbf{axiom} \vee \textbf{lemma} \vee \textbf{formula})\} && (2.7)\\
& \mathit{contents} \in \{(\textbf{limits} \vee \textbf{threshold}), (\textbf{gradient} \vee \textbf{calculus}), \textbf{proof}\}
\end{aligned}$$

V: *measure* $\in$ {**interval_scale, ratio_scale**} (2.8)
aggregation $\in$ {**values**, (**data_basis** $\vee$ **repository**), (**dashboard** $\vee$ **cockpit**)}

U: *type* $\in$ { **sociological_unit, economical_unit, physical_unit, hardware_unit, software_unit**} (2.9)
standard $\in$ {**non_standard, quasi_standard, standardized**}

E': *form:* see above (2.10)
extension $\in$ {**correction**, (**refinement** $\vee$ **approximation** $\vee$ **adaptation**), **extension**}

A': *domain:* see above (2.11)
changing $\in$ { **understood, improved, managed, controlled** }

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

MP: *approach* $\in$ {**aspect-oriented_measurement, capability-oriented_measurement, whole_measurement**} (2.12)

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.

2.2 Main Characteristics Preferences of Measurement Process Components

In the following we will present some examples of this kind of measurement aspects scaling.
Related to the measurement artefacts we can establish (note that the sign "$\prec$" characterizes the so-called ***evidence level*** (see [Kitchenham 2007])

$$A_{origin}^{aspects} \prec A_{origin}^{product \vee process \vee resources} \prec A_{origin}^{product \wedge process \wedge resources} . \quad (2.13)$$

Considering the measurement and including the application leads to

$$M_{sort}^{case\_study} \prec M_{sort}^{assessment} \prec M_{sort}^{improvement} \prec M_{sort}^{controlling} . \quad (2.14)$$

Addressing the tool aspects gives

$$T_{support}^{manual} \prec T_{support}^{semi\_automatic} \prec T_{support}^{automatic} . \quad (2.15)$$

Achieving the personnel background we obtain

$$P_{area}^{manager} \prec P_{area}^{researcher} \prec P_{area}^{practitioner} \, . \tag{2.16}$$

And finally addressing the used experiences leads to

$$E_{contents}^{principle} \prec E_{contents}^{analogy} \prec E_{contents}^{rules\_of\_thumb} \prec E_{contents}^{formula} \, . \tag{2.17}$$

2.3 Sub Characteristics Preferences of Measurement Process Components

Considering the sub characteristics we will present chosen relationships also. Relating to sub characteristics of the artefacts we can establish

$$A_{other\_pendant}^{domain} \prec A_{domain\_pendant}^{domain} \prec A_{original}^{domain} \, . \tag{2.18}$$

Describing the measurement and application aspects gives

$$M_{analogical\_conclusion}^{method} \prec M_{estimation}^{method} \prec M_{simulation}^{method} \prec M_{measurement}^{method} \, . \tag{2.19}$$

Relating the tool aspects leads to

$$T_{one\_meas.\_phase}^{level} \prec T_{some\_meas.\_phases}^{level} \prec T_{whole\_measurement}^{level} \, . \tag{2.20}$$

Achieving the personnel background as

$$P_{measurement\_expert}^{kind} \prec P_{application\_staff}^{kind} \, . \tag{2.21}$$

Furthermore, considering the experiences we obtain

$$E_{threshold}^{form} \prec E_{gradient}^{form} \prec E_{proof}^{form} \, . \tag{2.22}$$

2.4 Combined Characteristics Preferences of Measurement Process Components

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

$$A_{domain\_pendant}^{aspects} \prec A_{original}^{aspects} \prec A_{other\_pendant}^{product \vee process \vee resources} \prec A_{domain\_pendant}^{product \vee process \vee resources} \prec A_{other\_pendant}^{product \wedge process \wedge resources} \, . \tag{2.23}$$

or

$$M_{analogical\_conclusion}^{case\_study} \prec M_{estimation}^{improvement} \prec M_{estimation}^{controlling} \prec M_{simulation}^{experiment} \prec M_{simulation}^{assessment} \prec M_{measurement}^{case\_study} \prec M_{measurement}^{assessment} \prec M_{measurement}^{controlling} \, . \tag{2.24}$$

and

$$E_{threshold}^{law} \prec E_{calculus}^{law} \prec E_{proof}^{law} \prec E_{limits}^{analogy} \prec E_{gradient}^{analogy} \prec E_{proof}^{analogy} \quad (2.25)$$
$$\prec E_{threshold}^{criteria} \prec E_{calculus}^{rules\_of\_thumb} \prec E_{threshold}^{axiom} \prec E_{gradient}^{lemma} \prec E_{proof}^{formula}.$$

2.5 Simple Examples of Measurement Process Description

At first we will use our formal descriptions in order to describe some typical software measurement situations and implementations. Therefore we can establish some different levels of ***measurement evidence*** such as

- Using only the next lower levels of previous paradigm measurement experiences leads us 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 us to the ***measurement initialization***

In the following we will describe some examples using our scaled measurement process description. Usually, in the software development and application we can describe some of the following tasks and activities based on our formal background [Dumke 2007].

First general metrics application:

Our first example shows a simple (first) application of metrics based on a simple measurement process.

$$MP_{inhouse}^{aspect\_oriented}: \quad \mathbf{(2.26)}$$
$$(G_{internal\_goals}^{evaluation} \times A_{original}^{product\_aspects} \times M_{measurement}^{experiment})_{T_{some\_meas.\_phases}^{semi\_automatic}, P_{measurement\_expert}^{practitioner}}$$
$$\rightarrow (Q_{normalization}^{ordinal\_scale} \times E_{threshold}^{formulas})$$

Product quality assurance:

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

$$MP_{inhouse}^{capability-oriented}: \quad \mathbf{(2.27)}$$
$$(G_{external\_goals}^{managing} \times A_{original}^{product} \times M_{measurement}^{assessment})_{T_{measurement\_phases}^{semi\_automatic}, P_{measurement\_expert}^{practitioner}}$$
$$\rightarrow V_{cockpit}^{ratio\_scale} \times U_{standardized}^{software}$$

Process improvement:

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

$$MP_{inhouse}^{capability-oriented}: \quad \mathbf{(2.28)}$$
$$(G_{goals\_in\_use}^{improving} \times A_{original}^{process} \times M_{measurement}^{improvement})_{T_{measurement\_phases}^{semi\_automatic}, P_{measurement\_appl.\_staff}^{practitioner}}$$

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

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

Project controlling:

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

$$MP^{\text{capability}-\text{oriented}}_{\text{global\_production}}: \qquad \textbf{(2.29)}$$

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

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

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

Resources adaptation:

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

$$MP^{\text{capability}-\text{oriented}}_{\text{outsourced}}: \qquad \textbf{(2.30)}$$

$$(G^{\text{improving}}_{\text{goals\_in\_use}} \times A^{\text{resources}}_{\text{pendant}} \times M^{\text{improvement}}_{\text{measurement}})\, T^{\text{semi\_automatic}}_{\text{measurement\_phases}}, P^{\text{practitioner}}_{\text{measurement\_expert}}$$

$$\rightarrow (Q^{\text{identification}}_{\text{sinle\_value}} \times E^{\text{intuition}}_{\text{threshold}})\, T^{\text{semi\_automatic}}_{\text{measurement\_phases}}, P^{\text{practitioner}}_{\text{measurement\_expert}}$$

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

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 we must consider all the MP characteristics (at first the G level then the A level etc.). Hence we obtain

$$(2.26) \preccurlyeq (2.28) \preccurlyeq (2.30) \preccurlyeq (2.27) \preccurlyeq (2.29)$$

or

$$MP^{traditional}_{first\_metrics\_appl.} \preccurlyeq MP^{traditional}_{process\_improvement} \preccurlyeq MP^{traditional}_{resource\_adaptation} \preccurlyeq \qquad (2.31)$$

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

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

2.6 Measurement Process Improvements

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

$$MP^{\text{aspect\_oriented}}_{\text{outsourced}}: \qquad \textbf{(2.32)}$$

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

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

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

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

$$MP^{aspect\_oriented}_{inhouse}: \tag{2.33}$$

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

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

(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***

$$MP^{whole\_measurement}_{inhouse}: \tag{2.34}$$

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

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

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

Furthermore, we will differentiate the following ***graduation of measurement improvements*** 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 Software e-Measurement Processes as Ubiquitous Measurement

3.1 Basics of e-Measurement

In following we will give some examples of formal modelling of measurement processes embedded, oriented, involved and implemented in the World Wide Web. This kind of software measurement was called *e-Measurement* and was defined by Lother [Lother 2007] (see also [Abran 2006], [Ebert 2007], [Dumke 2004] and [Farooq 2008]) as:

> "*Software e-Measurement is the process of the quantification of object's or component's attributes according to selected measurement goals by using the capabilities of ICN (Information, Communication, Net) technologies.*"

Let us establish the basic components of the traditional software assurance characterized in the following figure by Lother [Lother 2007].

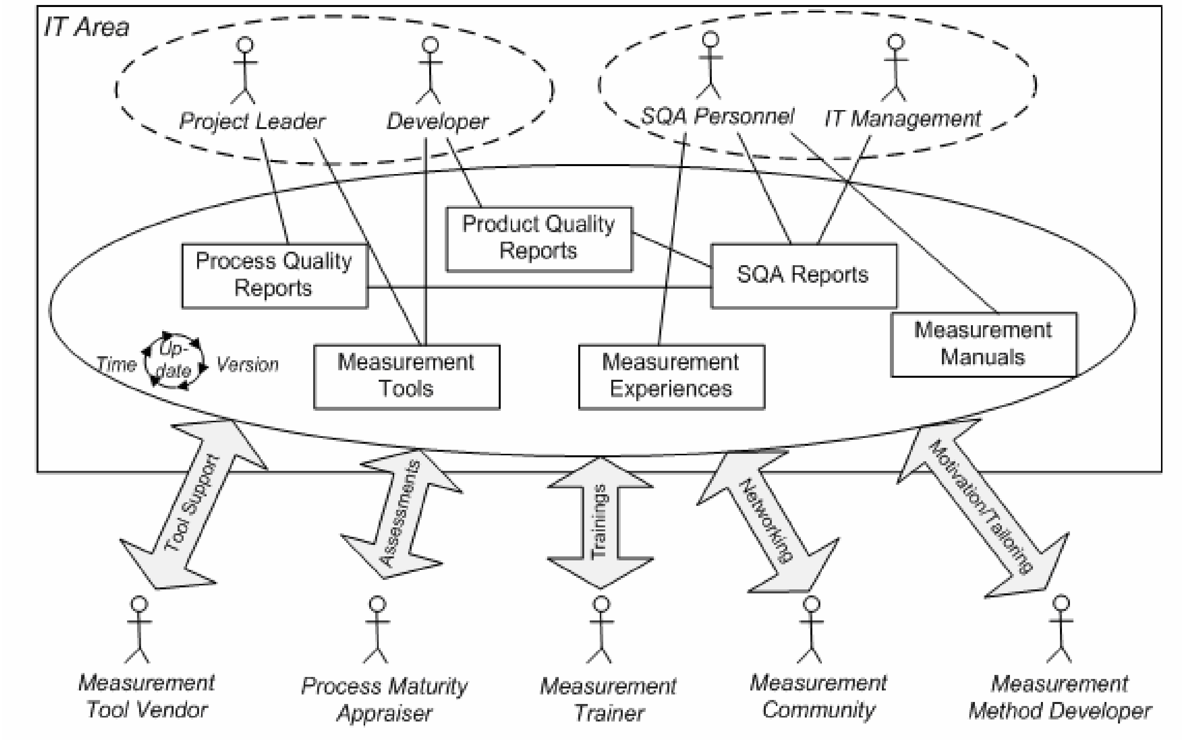


Figure 14: The traditional software assurance approach

Then the software e-Measurement could be described in the following manner (also adapted from [Lother 2007]) shown in the figure 15.

This e-Measurement can be divided in different kinds of measurement such as e-Measurement services, e-Measurement repositories etc. Note that especially the Web 2.0 hype can provide any new kinds of services, roles and infrastructures in the world-wide software quality assurance community and marketplaces.

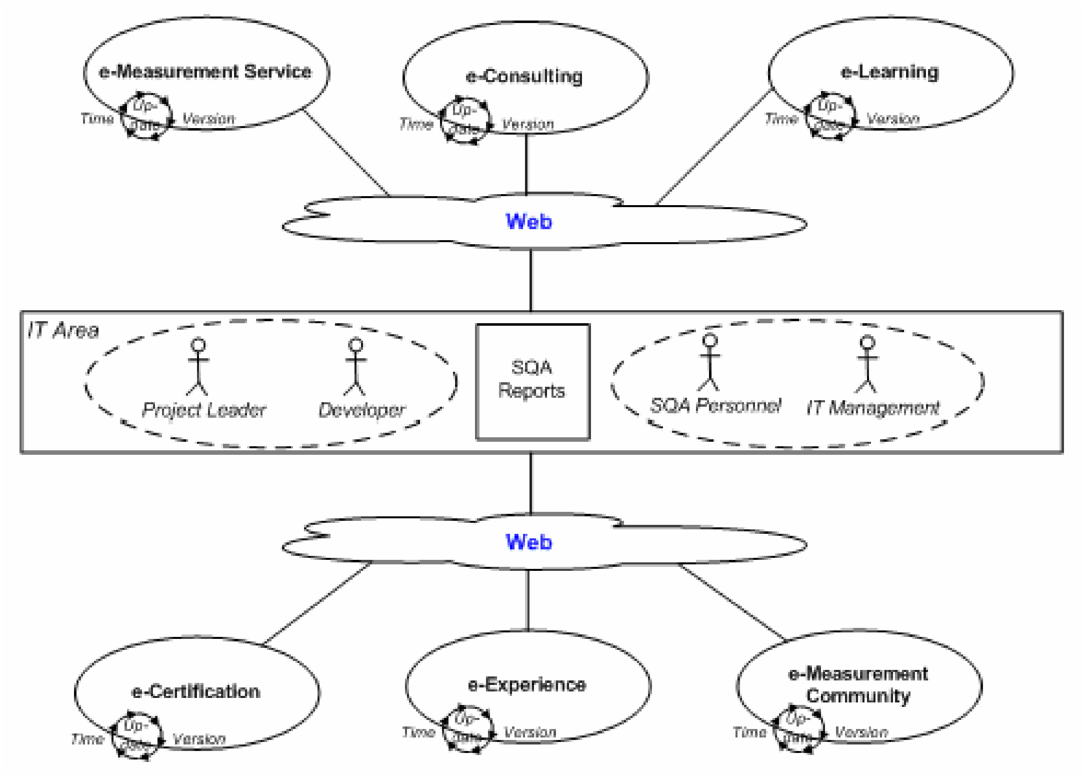


Figure 15: The software e-Measurement based quality assurance

In the following we will characterize some of these external (Web based) components in software quality assurance (based on measurement) and their achieved measurement levels.

3.2 Description of Chosen e-Measurement Processes

Note that the formal indexes in the following formal descriptions characterize the main kinds of Web technology.

The **e-Measurement Service** as a Web service usable for everyone can be described as

$$\boldsymbol{MP}^{e-Measurement}_{e-Service}: \qquad (3.1)$$

$$(\boldsymbol{G} \times \boldsymbol{A} \times \boldsymbol{M})_{T_{Web\_technology},\boldsymbol{P}} \rightarrow (\boldsymbol{Q} \times \boldsymbol{E})_{T_{Web\_technology},\boldsymbol{P}} \rightarrow (\boldsymbol{V} \times \boldsymbol{U})_{T_{Web\_technology},\boldsymbol{P}} \rightarrow \boldsymbol{E'} \times \boldsymbol{A'}$$

with a simple explanation as *e-Service* ∈ {*global_production, outsourced*} and *Web_technology* ∈ {*document-based, dynamic, semantic, service, mobile, agent, operational*}.

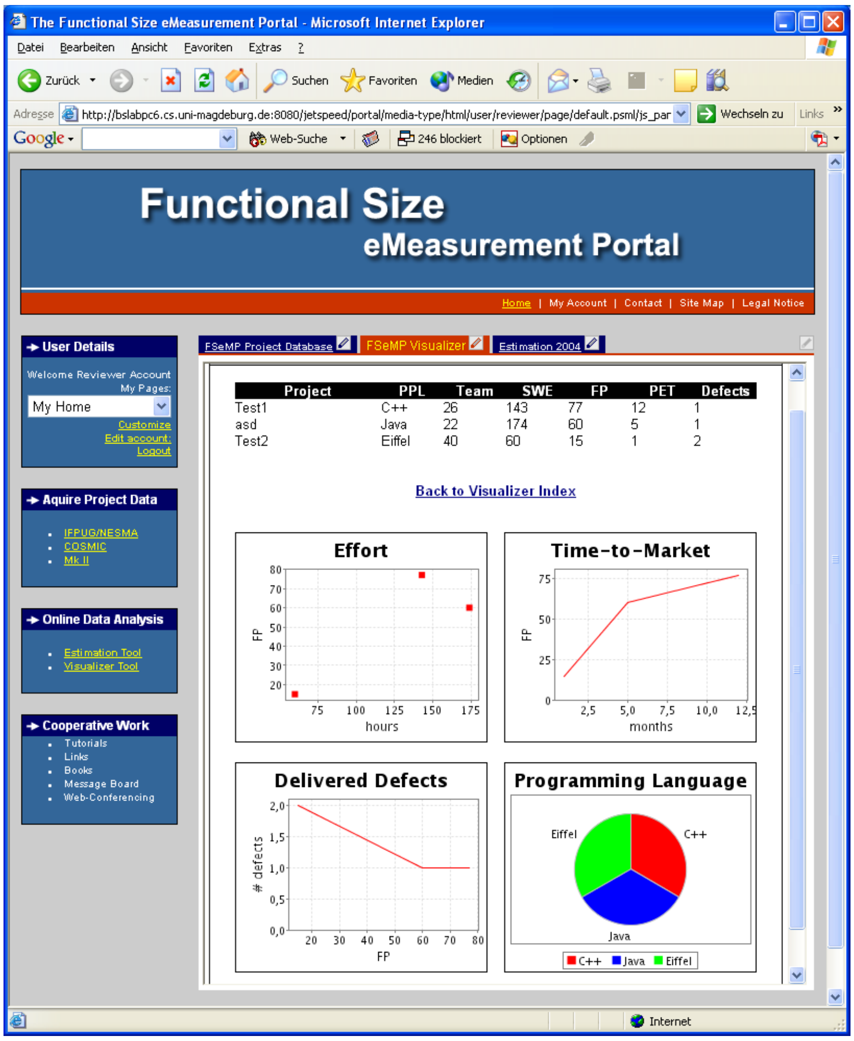
Note that we have shown such a service in figure 8 including both as measurement results and the measurement of Java applications.

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

$$
\begin{aligned}
\boldsymbol{MP}^{e-Measurement}_{e-Community} &: (\boldsymbol{G} \times \boldsymbol{A} \times \boldsymbol{M})\ T_{Web\_technology},\ P_{system\_operationality} \\
&\rightarrow (\boldsymbol{Q} \times \boldsymbol{E})\ T_{Web\_technology},\ P_{system\_operationality} \\
&\rightarrow (\boldsymbol{V} \times \boldsymbol{U})\ T_{Web\_technology}\ \ P_{system\_operationality} \rightarrow \boldsymbol{E}' \times \boldsymbol{A}'
\end{aligned}
\tag{3.2}
$$

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*}.

An example for the FSM community was implemented prototypically by Lother shown in the following figure [Lother 2004].



Project	PPL	Team	SWE	FP	PET	Defects
Test1	C++	26	143	77	12	1
asd	Java	22	174	60	5	1
Test2	Eiffel	40	60	15	1	2

Figure 16: The Functional Size Measurement community portal

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

$$MP^{e-Measurement}_{e-Experience} : (\boldsymbol{G} \times \boldsymbol{A} \times \boldsymbol{M})\; T_{Web\_technology}\, , P_{system\_operationality} \tag{3.3}$$

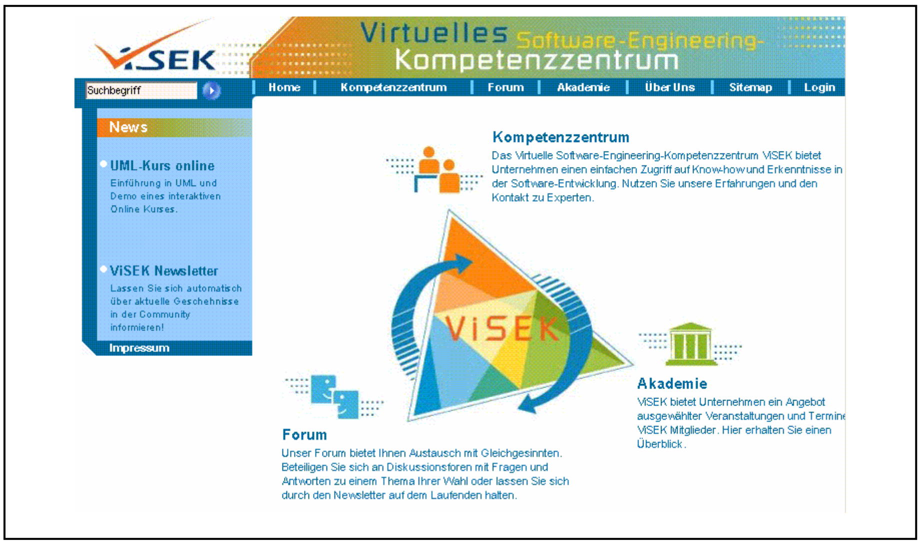
$$\rightarrow (\boldsymbol{Q}_{Web\_technology} \times \boldsymbol{E}_{Web\_technology})\, T_{Web\_technology}\, , P_{system\_operationality}$$

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

$$\rightarrow \boldsymbol{E'}_{Web\_technology} \times \boldsymbol{A'}$$

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*}.

An example of Web-based services of experiences is shown in the following figure including descriptions of software engineering methods and practices (http://www.software-kompetenz.de).



UML-Kurs online
Einführung in UML und Demo eines interaktiven Online Kurses.

ViSEK Newsletter
Lassen Sie sich automatisch über aktuelle Geschehnisse in der Community informieren!

Kompetenzzentrum
Das Virtuelle Software-Engineering-Kompetenzzentrum ViSEK bietet Unternehmen einen einfachen Zugriff auf Know-how und Erkenntnisse in der Software-Entwicklung. Nutzen Sie unsere Erfahrungen und den Kontakt zu Experten.

Akademie
ViSEK bietet Unternehmen ein Angebot ausgewählter Veranstaltungen und Termine ViSEK Mitglieder. Hier erhalten Sie einen Überblick.

Forum
Unser Forum bietet Ihnen Austausch mit Gleichgesinnten. Beteiligen Sie sich an Diskussionsforen mit Fragen und Antworten zu einem Thema Ihrer Wahl oder lassen Sie sich durch den Newsletter auf dem Laufenden halten.

Figure 17: The German software engineering experience portal

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

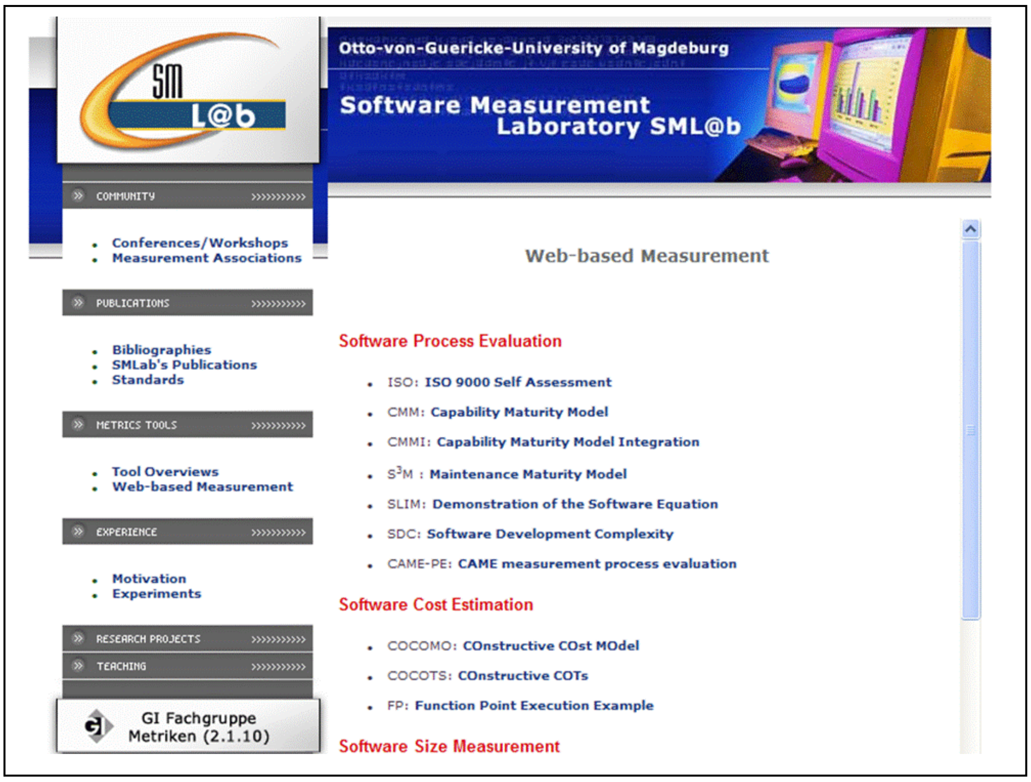
$$MP^{e-Measurement}_{e-Quality} : (\boldsymbol{G} \times \boldsymbol{A} \times \boldsymbol{M}_{Web\_technology})\, T_{Web\_technology}\, , P_{system\_operationality} \tag{3.4}$$

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

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

$$\rightarrow \boldsymbol{E'}_{Web\_technology} \times \boldsymbol{A'}$$

with an explanation as *e-Quality* $\in$ {*information, certification, consulting, estimation*}, *Web_technology* $\in$ {*document-based, dynamic, semantic, service, mobile, agent, operational*} and *system_operationality* $\in$ {*coordination, conferencing, cooperation, collaboration*}.
The SML@b Web application could be considered as example quality services by using existing (estimation) methods.


Otto-von-Guericke-University of Magdeburg

Software Measurement Laboratory SML@b

COMMUNITY
- Conferences/Workshops
- Measurement Associations

PUBLICATIONS
- Bibliographies
- SMLab's Publications
- Standards

METRICS TOOLS
- Tool Overviews
- Web-based Measurement

EXPERIENCE
- Motivation
- Experiments

RESEARCH PROJECTS

TEACHING

GI Fachgruppe Metriken (2.1.10)

Web-based Measurement

Software Process Evaluation
- ISO: ISO 9000 Self Assessment
- CMM: Capability Maturity Model
- CMMI: Capability Maturity Model Integration
- S^{3}M : Maintenance Maturity Model
- SLIM: Demonstration of the Software Equation
- SDC: Software Development Complexity
- CAME-PE: CAME measurement process evaluation

Software Cost Estimation
- COCOMO: COnstructive COst MOdel
- COCOTS: COnstructive COTs
- FP: Function Point Execution Example

Software Size Measurement

Figure 18: The quality method application in the SML@b

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

$$MP^{e-Measurement}_{e-Control}: \tag{3.5}$$

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

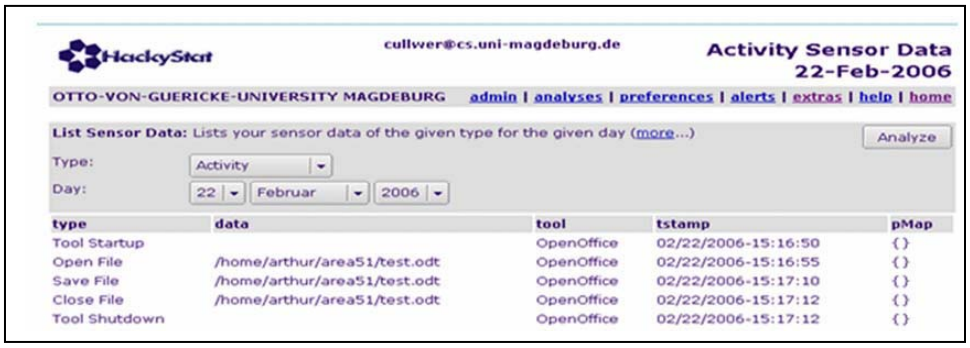
$$\rightarrow (\boldsymbol{Q}_{Web\_technology} \times \boldsymbol{E}_{Web\_technology})\, T_{Web\_technology}, P_{system\_operationality}$$

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

$$\rightarrow \boldsymbol{E'}_{Web\_technology} \times \boldsymbol{A'}$$

with the details as *e-Control* $\in$ {*evaluation, improvement, managing, controlling*}, *Web_technology* $\in$ {*document-based, dynamic, semantic, service, mobile, agent, operational*}, *system_operationality* $\in$ {*coordination, conferencing, cooperation, collaboration*} and *type_of_measurement* $\in$ {*modelling, measurement, evaluation, application*}.

A simple example of process controlling is given in the following figure that extends any office solutions in order to measure the different files using profiles (see [Abran 2006]).


HackyStat

cullwer@cs.uni-magdeburg.de

Activity Sensor Data
22-Feb-2006

OTTO-VON-GUERICKE-UNIVERSITY MAGDEBURG   admin | analyses | preferences | alerts | extras | help | home

List Sensor Data: Lists your sensor data of the given type for the given day (more...)   Analyze

Type: Activity

Day: 22 Februar 2006

type	data	tool	tstamp	pMap
Tool Startup		OpenOffice	02/22/2006-15:16:50	{}
Open File	/home/arthur/area51/test.odt	OpenOffice	02/22/2006-15:16:55	{}
Save File	/home/arthur/area51/test.odt	OpenOffice	02/22/2006-15:17:10	{}
Close File	/home/arthur/area51/test.odt	OpenOffice	02/22/2006-15:17:12	{}
Tool Shutdown		OpenOffice	02/22/2006-15:17:12	{}

Figure 19: The HackyStat extension for process controlling

Finally, the **Measurement e-Learning** as one of the measurement training aspects can be formalized as

$$\boldsymbol{MP}_{e-Learning}^{e-Measurement}: \tag{3.6}$$

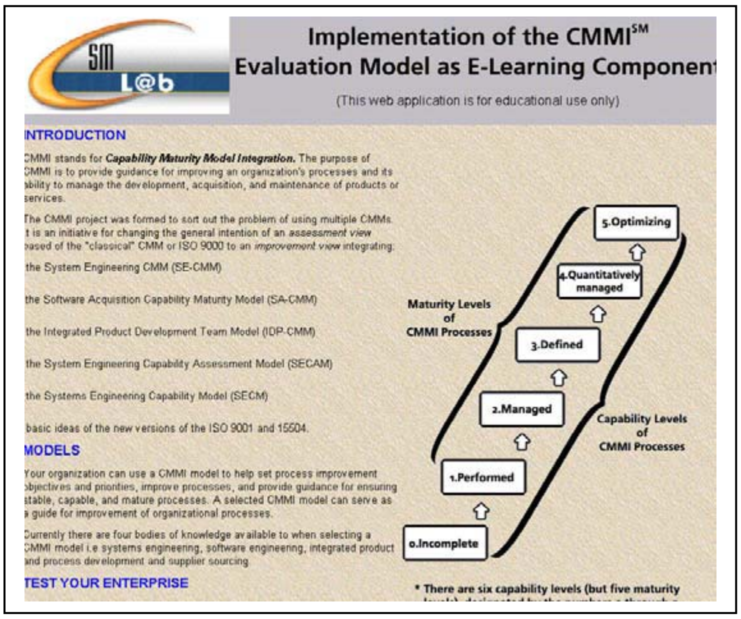
$$(\boldsymbol{G}_{Web\_technology} \times \boldsymbol{A}_{Web\_technology} \times \boldsymbol{M}_{Web^{measurement\_operation}})_{T_{Web\_technology}}, P_{system\_operationality}$$

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

$$\to (\boldsymbol{V}_{Web\_technology} \times \boldsymbol{U}_{Web\_technology})_{T_{Web\_technology}}, P_{system\_operationality}$$

$$\to \boldsymbol{E'}_{Web\_technology} \times \boldsymbol{A'}_{Web\_technology}$$

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 following example of CMMI application in the Web demonstrates the measurement e-Learning in principle.


Implementation of the CMMI^{SM} Evaluation Model as E-Learning Componen

(This web application is for educational use only)

INTRODUCTION

CMMI stands for ***Capability Maturity Model Integration.*** The purpose of CMMI is to provide guidance for improving an organization's processes and its bility to manage the development, acquisition, and maintenance of products or services.

The CMMI project was formed to sort out the problem of using multiple CMMs. t is an initiative for changing the general intention of an *assessment view* based of the "classical" CMM or ISO 9000 to an *improvement view* integrating:

the System Engineering CMM (SE-CMM)

the Software Acquisition Capability Maturity Model (SA-CMM)

the Integrated Product Development Team Model (IDP-CMM)

the System Engineering Capability Assessment Model (SECAM)

the Systems Engineering Capability Model (SECM)

basic ideas of the new versions of the ISO 9001 and 15504.

MODELS

Your organization can use a CMMI model to help set process improvement objectives and priorities, improve processes, and provide guidance for ensuring stable, capable, and mature processes. A selected CMMI model can serve as a guide for improvement of organizational processes.

Currently there are four bodies of knowledge available to when selecting a CMMI model i.e systems engineering, software engineering, integrated product and process development and supplier sourcing.

TEST YOUR ENTERPRISE

* There are six capability levels (but five maturity

Figure 20: The CMMI explanation and application in the SML@b

Note that this kind of formalization motivates further ideas and possibilities of Web-based software measurement supports and innovations (examples are given in [Abran 2006], [Dumke 2003a], [Dumke 2004], [Ebert 2007], [Lother 2004] and [Lother 2007]).

3.3 Measurement Levels in e-Measurement

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.

$$MP^{aspect\_oriented}_{outsourced} : (G^{evaluation}_{internal\_goals} \times A^{process}_{original} \times M^{assessment}_{estimation})T^{semi\_automatic}_{one\_meas.\_phases}, P^{practitioner}_{meas.\_expert\_staff} \rightarrow (Q^{ordinal\_scale}_{sinle\_value} \times E^{analogy}_{threshold}) \quad (3.7)$$

Another example of measurement e-Learning based on the “Web-based Measurement” at the SML@b, http://www.smlab.de) as *Java measurement service* has the following measurement characteristics (as immediate measurement level also).

$$MP^{aspect\_oriented}_{outsourced} : (G^{evaluation}_{internal\_goals} \times A^{product\_aspects}_{original} \times M^{assessment}_{measurement})T^{semi\_automatic}_{some\_meas.\_phases}, P^{researcher}_{meas.\_expert\_staff} \rightarrow (Q^{ordinal\ scale}_{single\_value} \times E^{intuition}_{threshold}) \quad (3.8)$$

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.

$$MP^{capability\_oriented}_{outsourced} : (G^{managing}_{goals\_in\_use} \times A^{resources}_{original} \times M^{controlling}_{measurement})T^{automatic}_{whole\_measurement}, P^{practitioner}_{meas.\_appl.\_staff} \rightarrow (V^{ratio\_scale}_{cockpit} \times U^{software\_unit}_{standardized})\, T^{automatic}_{whole\_measurement}, P^{practitioner}_{meas.\_appl.\_staff} \rightarrow E^{formula}_{extension} \times A^{resources}_{controlled} \quad (3.9)$$

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

$$MP^{traditional}_{first\_metrics\_appl.} \precsim MP^{e-Measurement}_{e-Service} \precsim MP^{traditional}_{product\_quality\_assurance.} \quad (3.10)$$

and

$$MP^{e-Measuremen\ t}_{e-Control} \precsim MP^{traditional}_{project\_controlling} \quad (3.11)$$

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.

4 Measurement as Controlling for Agent-Based and Self-Managed Systems

4.1 Characteristics of Agent-Oriented Software Engineering (AOSE)

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 [Bauer 2004], [Ciancarini 2001], [Gerber 2001], [Huhns 2004], [Jennings 1998], [Knapik 1998], [Liu 2001], [Panait 2006] and [Wooldridge 2002]).

Software agents: The essential components of a software agent form a measurement point of view in the following scheme [Wille 2005].

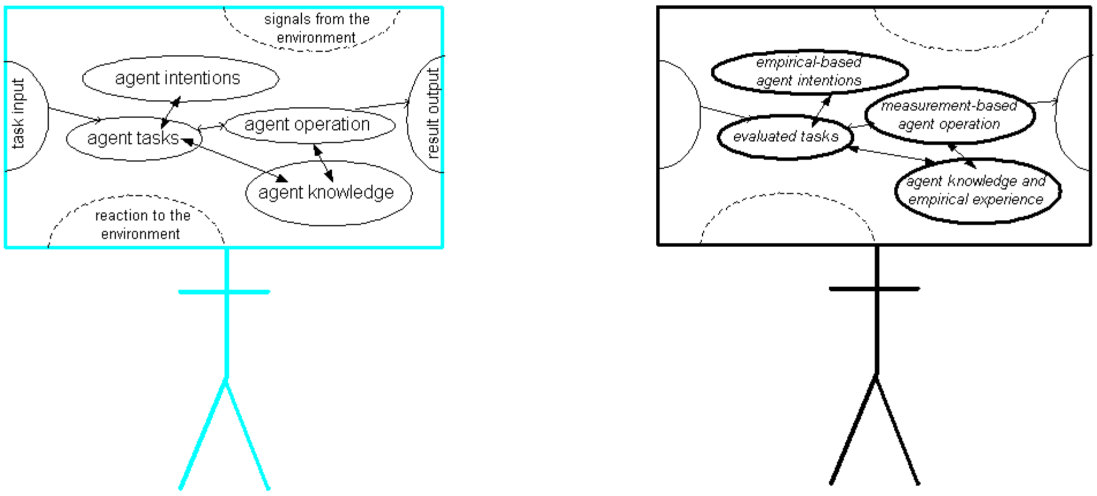

signals from the environment
agent intentions
task input
result output
agent tasks
agent operation
agent knowledge
reaction to the environment
empirical-based agent intentions
measurement-based agent operation
evaluated tasks
agent knowledge and empirical experience

Figure 21: Components of a general software agent (original and measured)

The next aspects of software agents are related to their *communication and/or co-operation.* The following figure explains these aspects in general.

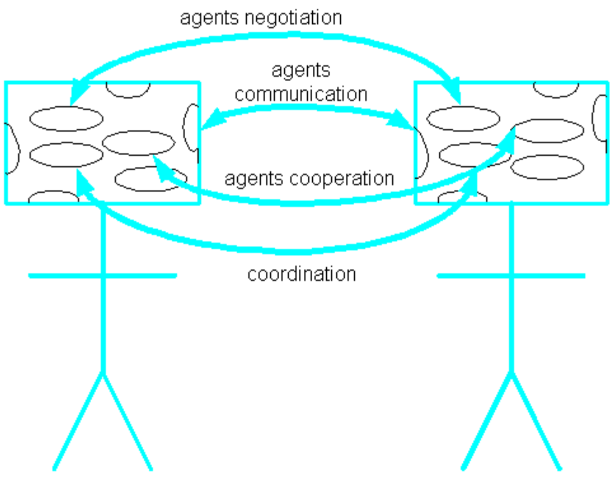


Figure 22: Communications between software agents

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.

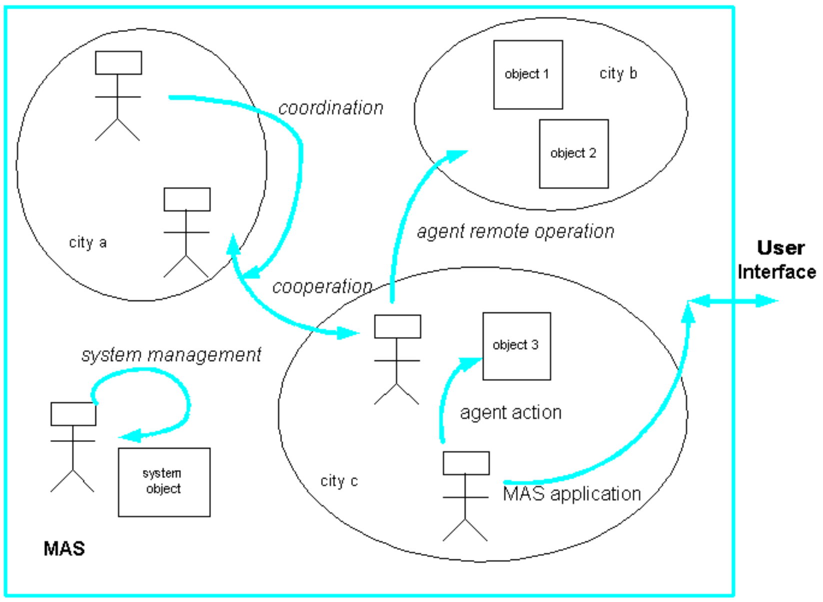


Figure 23: General components of multi-agent systems (MAS)

The following figure suggests a all measurement intentions for our agent-based systems.

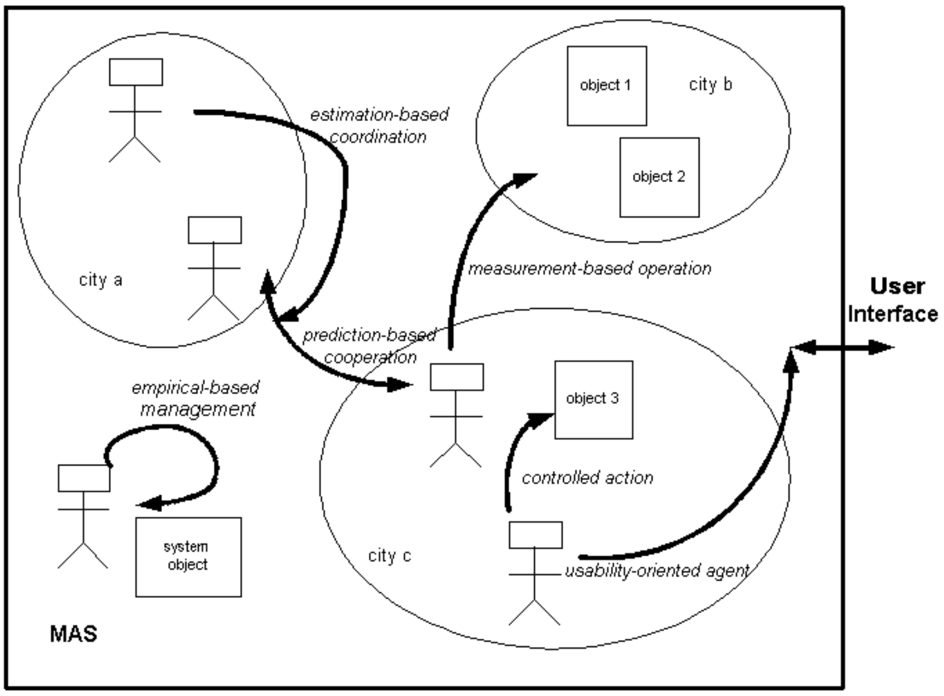
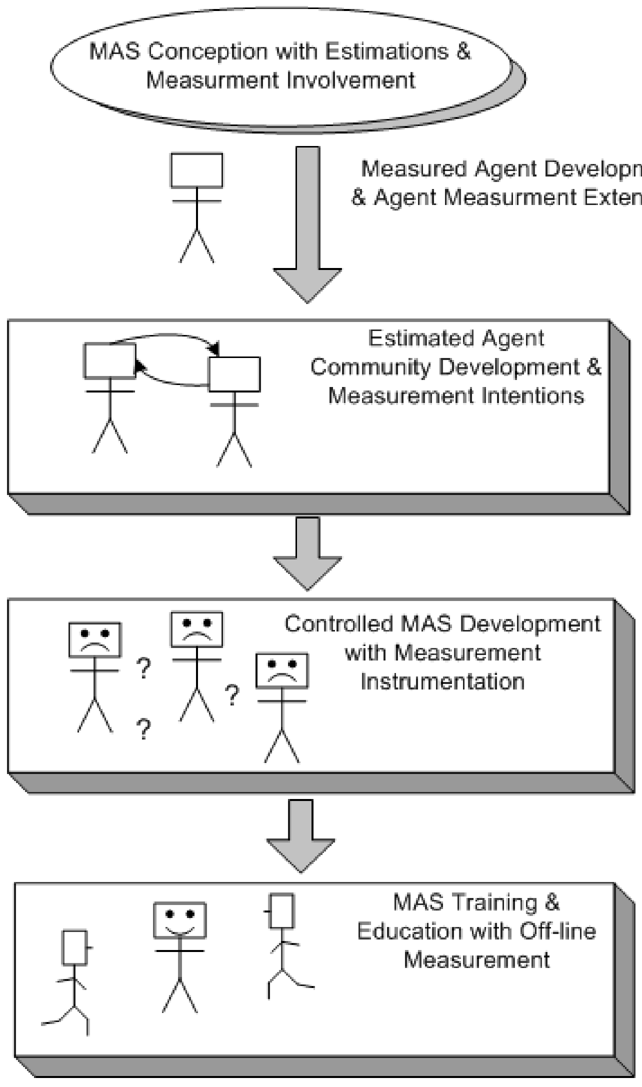


Figure 24: Measurement-based MAS architecture

MAS development: The specification, design and implementation of agent-based system and/or MAS differs 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 [Mencke 2007].



MAS Conception with Estimations & Measurment Involvement
Measured Agent Development & Agent Measurment Extentions
Estimated Agent Community Development & Measurement Intentions
Controlled MAS Development with Measurement Instrumentation
?
?
?
MAS Training & Education with Off-line Measurement

Figure 25: Measurement-based MAS development

4.2 AOSE related Measurement Extensions

First, we describe the measurement of software agents considering the new kind of controlling by the agents themselves.

$$MP^{AOSE}_{agents} : \tag{4.1}$$

$$(G^{managing}_{agent\_intention} \times A^{agents}_{original} \times M^{controlling}_{measurement})T^{automatic}_{some\_measurement}$$
$$\rightarrow (V^{ratio\_scale}_{repository} \times U^{software\_unit}_{quasi\_standard})\,T^{automatic}_{some\_measurement}$$
$$\rightarrow E^{agent\_knowledge\_basis}_{extension} \times A^{improved\_agent}_{controlled}$$

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

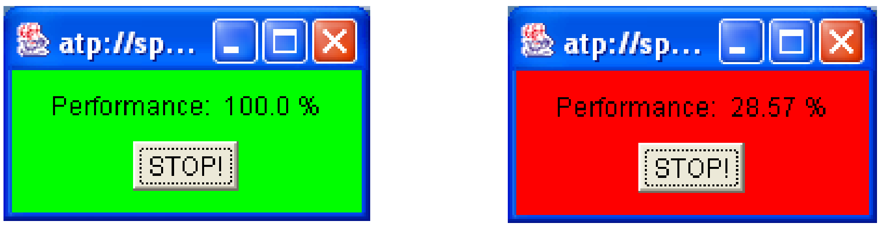
$$measurement \in \{ \mu_{agent}^{size}, \mu_{agent}^{structure}, \mu_{agent}^{complexity}, \mu_{agent}^{functionality}, \mu_{agent}^{description(development)}, \mu_{agent}^{description(application)}, \mu_{agent}^{description(publication)}, \mu_{agent}^{communicaton}, \mu_{agent}^{interaction}, \mu_{agent}^{learning}, \mu_{agent}^{adaptation}, \mu_{agent}^{negotiation}, \mu_{agent}^{collaboration}, \mu_{agent}^{coordination}, \mu_{agent}^{cooperation}, \mu_{agent}^{reproduction}, \mu_{agent}^{performance}, \mu_{agent}^{specialization} \} \quad (4.2)$$

The metrics for the ***agent design level*** are: *Software agent size* μ_{agent}^{size}: The size considers both aspects of an agent: the functional size and the physical size of a software agent. *Software agent component structure* $\mu_{agent}^{structure}$: The structure depends on the kind of the agent (intelligent, reactive, deliberative etc.), and the agent interface is related to the kind of agent coupling (as fixed, variable or evolutionary). *Software agent complexity* $\mu_{agent}^{complexity}$: The complexity is divided into the computational and psychological complexity and should be measured using both concrete aspects. *Software agent functionality* $\mu_{agent}^{functionality}$: This aspect considers the appropriateness of the agent with respect to the requirements.

The metrics for the ***agent description level*** are: *Software agent development description level* $\mu_{agent}^{description(development)}$: It considers the completeness of the development documentation (including tests and change supports). *Software agent application description level* $\mu_{agent}^{description(application)}$: The metric includes the quality (readability, completeness, on-line support etc.) of the user documentation. *Software agent publication description level* $\mu_{agent}^{description(publication)}$: This metric considers the public relations for using the software agent and involves the system description.

The metrics for the ***agent working level*** are: *Software agent communication level* $\mu_{agent}^{communicaton}$: Considers of the size of communication and the level of the conversation required to sustain the activities. *Software agent interaction level* $\mu_{agent}^{interaction}$: This metric is related to the agent context and environment and their different kinds of actions (as transformation, reflecting, executing, modification, commands, perception, deliberation). *Software agent learning level* $\mu_{agent}^{learning}$: This metric evaluates the skills, intentions, and actions of extending the agent facilities itself. *Software agent adaptation level* $\mu_{agent}^{adaptation}$: The adaptation metric considers facilities of an agent changing in order to react to new conditions in the environment. *Software agent negotiation level* $\mu_{agent}^{negotiation}$: The measurement is based on the evaluation of facilities like the agent intentions, conflict resolution, and realized commitments for successful negotiation. *Software agent collaboration* *level* $\mu_{agent}^{collaboration}$: This metric is oriented towards the agent's facility to work together with other agents. *Software agent coordination level* $\mu_{agent}^{coordination}$: The agent's facility of managing any one agent task is considered. *Software agent cooperation level* $\mu_{agent}^{cooperation}$: This metric considers the volume and efficiency of an agent relating to a common task. *Software agent self-reproduction level* $\mu_{agent}^{reproduction}$: The number of destroyed agents related to repaired agents is counted. *Software agent performance level* $\mu_{agent}^{performance}$**:** This metric considers the task related performance of an agent. *Software agent specialization level* $\mu_{agent}^{specialization}$: The metric considers the degree of specialization and the degree of redundancy of an agent.

Note that the metrics-based analysis of the agent behavior is one of the new and extended areas in software measurement of agent-based systems. An example of agent measurement is shown in the following figure (left the green, right the red evaluation based on continued measurement [Wille 2005]).


atp://sp...

Performance: 100.0 %

STOP!

atp://sp...

Performance: 28.57 %

STOP!

Figure 26: Examples of agent measurement (as aglet performance)

We describe the measurement of multi-agent systems (MAS) considering the new kind of system controlling in the same manner as.

$$MP_{MAS}^{AOSE}: \tag{4.3}$$

$$(G_{user\_intention}^{managing} \times A_{original}^{MAS} \times M_{measurement}^{controlling})\,T_{some\_measurement}^{automatic}$$

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

$$\rightarrow E_{extension}^{agent\_knowledge\_basis} \times A_{controlled}^{improved\_MAS}$$

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

$$measurement \in \{\, \mu_{MAS}^{size}, \mu_{MAS}^{structure}, \mu_{MAS}^{complexity}, \mu_{MAS}^{functionality}, \tag{4.4}$$

$$\mu_{MAS}^{description(development)}, \mu_{MAS}^{description(application)}, \mu_{MAS}^{description(publication)},$$

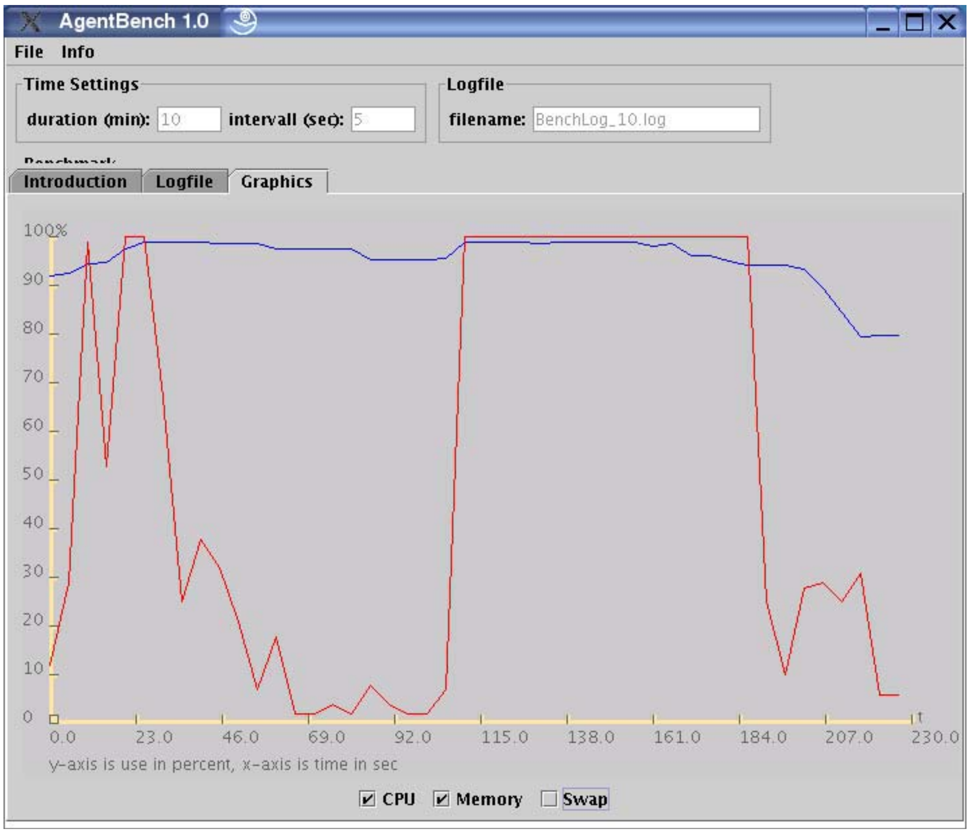
$$\mu_{MAS}^{communication}, \mu_{MAS}^{interaction}, \mu_{MAS}^{knowledge}, \mu_{MAS}^{lifeness}, \mu_{MAS}^{conflict}, \mu_{MAS}^{community},$$

$$\mu_{MAS}^{management}, \mu_{MAS}^{application}, \mu_{MAS}^{stability}, \mu_{MAS}^{performance}, \mu_{MAS}^{organization}\}$$

The metrics for the ***MAS design level***: *Agent system size* μ_{MAS}^{size}: The measured system size includes the potential number of (active) agents and their contents; further, the size is related to the environment. *Agent system component structure* $\mu_{MAS}^{structure}$: This metric includes agent the type of organizational structure (hierarchies or egalitarian), the degree of parallelism, the kinds of organizational functions (representational, organizational, conative, interactional, productive, or preservative). *Agent system complexity* $\mu_{MAS}^{complexity}$: One of these measured aspects leads to the degree of the organizational dimensions (social, relational, physical, environmental, and personal). *Agent system functionality* $\mu_{MAS}^{functionality}$: This metric considers the realization of all of the functional system requirements.

The metrics for the ***MAS description level***: *Agent system development description level* $\mu_{MAS}^{description(development)}$: This metric considers the integration of the agent concepts and dynamics and their sufficient documentation. *Agent system application description level* $\mu_{MAS}^{description(application)}$: This considers the user documentation of all aspects of the system applications related to the different user categories. *Agent system publication description level* $\mu_{MAS}^{description(publication)}$: Publication metrics evaluate the user acceptance and marketing aspects of the agent-based system application.

The metrics for the ***MAS working level:*** *Agent system communication level* $\mu_{MAS}^{communication}$: The number of ACLs between the different kinds of software agents and their different roles and actions. *Agent system interaction level* $\mu_{MAS}^{interaction}$: This metric deals with the average types of interactions relating to the agents and their roles in the environment of the agent-based system. *Agent system knowledge level* $\mu_{MAS}^{knowledge}$: This metric measures the results of agent learning for agent-based system (based on the different kinds of agents, either tropistic or hysteretic). *Agent system lifeness level* $\mu_{MAS}^{lifeness}$: This metric is based on the agent adaptation which reflects the adaptation level of the whole agent-based system. *Agent system conflict management level* $\mu_{MAS}^{conflict}$: The system success is based on agent negotiation and considers the relations between the different kinds of a fair play in the realization of the system tasks. *Agent system community level* $\mu_{MAS}^{community}$: This metric considers the level of different agent communities based on the agent collaboration. *Agent system management level* $\mu_{MAS}^{management}$: This system metric is based on the agent coordination level with respect to the whole agent system structure. *Agent system application level* $\mu_{MAS}^{application}$: This metric is related to the application area and the different agent roles in cooperation. *Agent system stability level* $\mu_{MAS}^{stability}$: The stability measure is based on the agent self-reproduction. *Agent system performance level* $\mu_{MAS}^{performance}$: The handling with object to realize special tasks through the different agents is considered. *Agent system organization level* $\mu^{organization}_{MAS}$: The different agent roles (archivist, customer, mediator, planner, decision-maker, observer, and communicator) are considered.


Time Settings

duration (min): 10 intervall (sec): 5

Logfile

filename: BenchLog_10.log

☑ CPU ☑ Memory ☐ Swap

Figure 27: Example of MAS measurement (as benchmarking [Gerber 2001])

Finally, we describe the measurement of MAS development including their resources in the same manner. Note that the MAS development consists of two pairs as the development of the agent(s) and the development/building of the MAS itself.

$$MP^{AOSE}_{MAS\_development}: \tag{4.5}$$

$$(G^{managing}_{customer\_intention} \times A^{agent\_system\_development}_{concept} \times M^{improvement}_{measurement})\, T^{semi\_automatic}_{some\_measurement}, P^{practitioner}_{measurement\_application\_staff}$$

$$\rightarrow (Q^{ordinal\_scale}_{repository} \times E^{criteria}_{threshold})\, T^{semi\_automatic}_{some\_measurement}, P^{practitioner}_{measurement\_application\_staff}$$

$$\rightarrow E^{methodology}_{extension} \times A^{implemented\_agent\_system}_{to\_be\_trained}$$

Especially, the measurement methods $\mu_{development(agent)}$ and $\mu_{development(MAS)}$ can be summarized as

$$measurement \in \{ \mu^{phases}_{development(Agent)}, \mu^{milestones}_{development(Agent)}, \mu^{workflow}_{development(Agent)}, \tag{4.6}$$

$$\mu^{methodology}_{development(Agent)}, \mu^{paradigm}_{development(Agent)}, \mu^{CASE}_{development(Agent)}, \mu^{management(project)}_{development(Agent)},$$

$\mu^{management(configuation)}_{development(Agent)}$, $\mu^{management(quality)}_{development(Agent)}$, $\mu^{phases}_{development(MAS)}$, $\mu^{milestones}_{development(MAS)}$, $\mu^{workflow}_{development(MAS)}$, $\mu^{methodology}_{development(MAS)}$, $\mu^{paradigm}_{development(MAS)}$, $\mu^{CASE}_{development(MAS)}$, $\mu^{management(project)}_{development(MAS)}$, $\mu^{management(configuration)}_{development(MAS)}$, $\mu^{management(quality)}_{development(MAS)}$, $\mu^{skill(developer)}_{development(Agent)}$, $\mu^{communication(developer)}_{development(Agent)}$, $\mu^{productivity(developer)}_{development(Agent)}$, $\mu^{paradigm(software)}_{development(Agent)}$, $\mu^{performance(software)}_{development(Agent)}$, $\mu^{replacement(software)}_{development(Agent)}$, $\mu^{reliability(hardware)}_{development(Agent)}$, $\mu^{performance(hardware)}_{development(Agent)}$, $\mu^{availability(hardware)}_{development(Agent)}$, $\mu^{skill(developer)}_{development(MAS)}$, $\mu^{comunication(developer)}_{development(MAS)}$, $\mu^{productivity(developer)}_{development(MAS)}$, $\mu^{paradigm(software)}_{development(MAS)}$, $\mu^{performance(software)}_{development(MAS)}$, $\mu^{replacement(software)}_{development(MAS)}$, $\mu^{reliability(hardware)}_{development(MAS)}$, $\mu^{performance(hardware)}_{development(MAS)}$, $\mu^{availability(hardware)}_{development(MAS)}$ }

The metrics for the ***agent development life cycle***: *Software agent phases level* $\mu^{phases}_{development(Agent)}$: The characteristics (size, structure, complexity) in the different development phases are considered. *Software agent milestones level* $\mu^{milestones}_{development(Agent)}$: This metric evaluates agent development with respect to a milestone. *Agent requirements workflow level* $\mu^{workflow}_{development(Agent)}$: This metric considers the implemented requirements during the development phases.

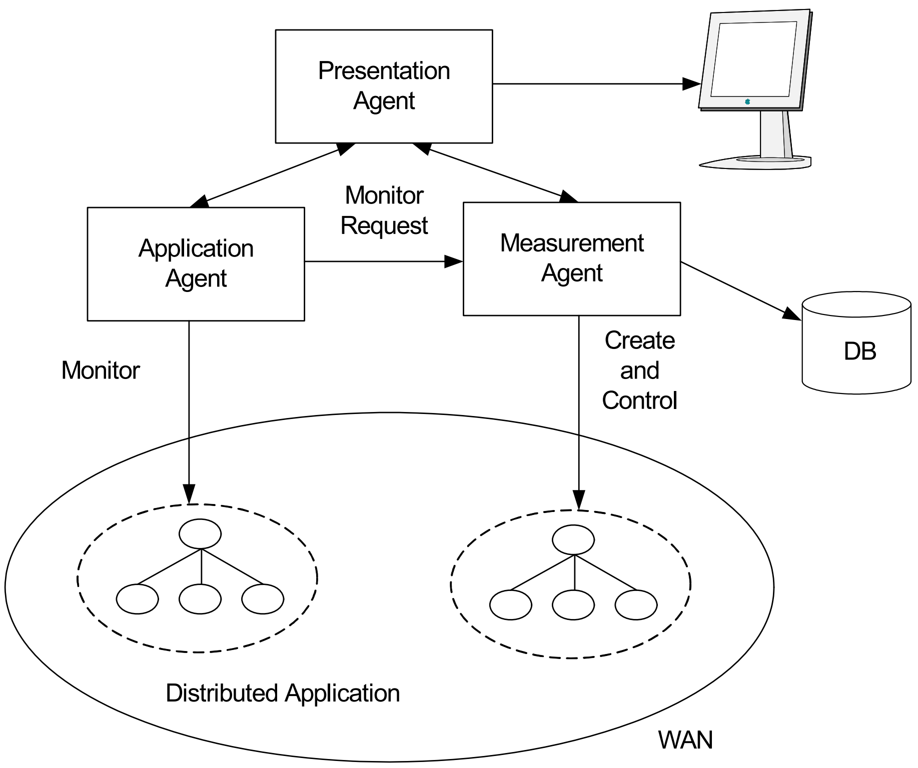


Figure 28: MAS development measurement as JAVALite extension [Wijata 2000]

The metrics for the ***agent development method level***: *Software agent methodology level* $\mu^{methodology}_{development(Agent)}$: The level of the development method used is quantified. *Software agent paradigm level* $\mu^{paradigm}_{development(Agent)}$: This metric evaluates the appropriateness of the chosen development paradigm. *Software agent CASE level* $\mu^{CASE}_{development(Agent)}$: This metric quantifies the tool support for the agent implementation.

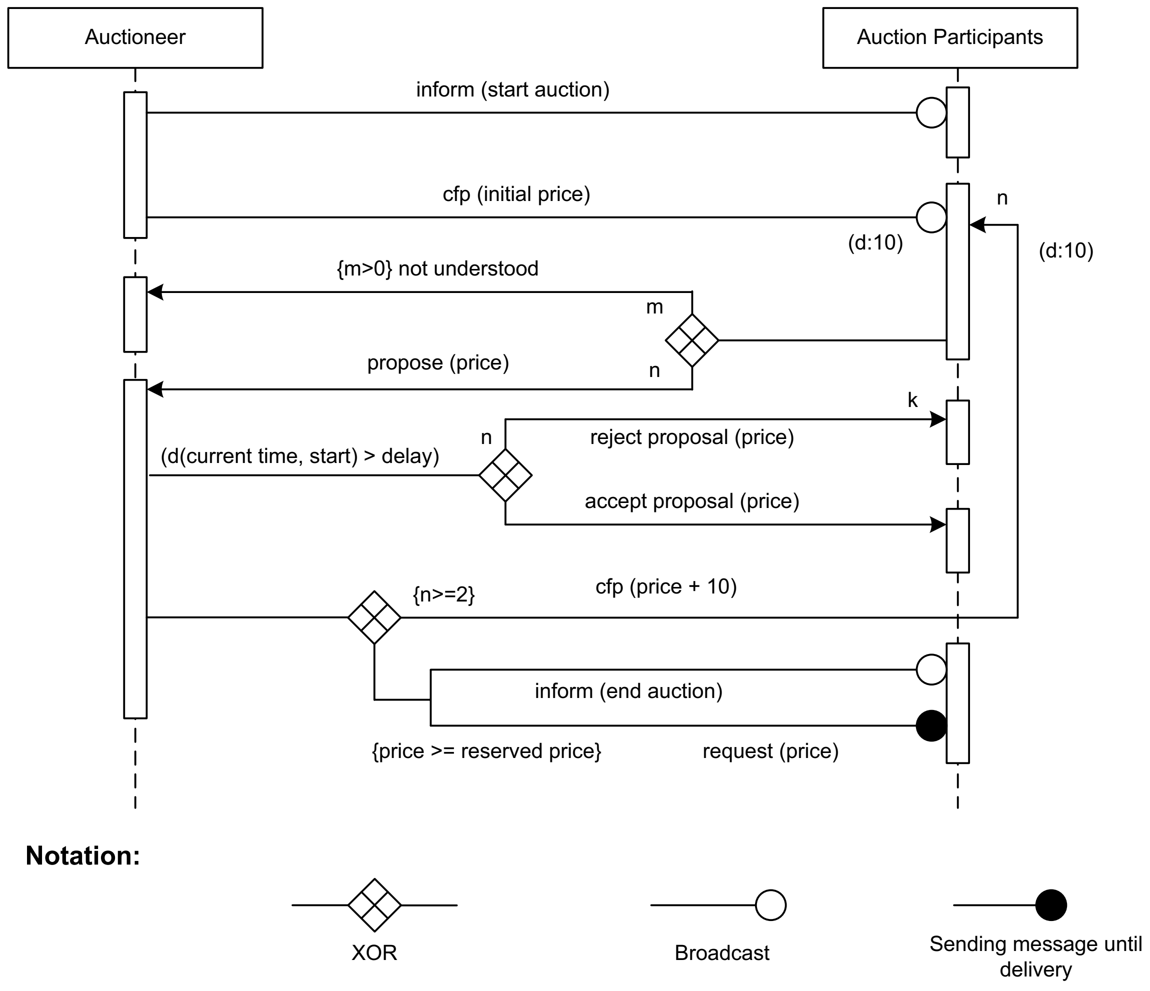


Figure 29: Example of Agent UML (AUML) application [Huhns 2004]

The metrics for the ***agent development management level***: *Agent project management level* $\mu^{management(project)}_{development(Agent)}$: This set of metrics considers the management level of the development risks and methods. *Agent configuration management level* $\mu^{management(configuation)}_{development(Agent)}$: This considers the successful of the version control with respect to an agent. *Agent quality management level* $\mu^{management(quality)}_{development(Agent)}$: This set of metrics considers the quality assurance techniques related to an agent.

Now, we consider the development process of the MAS and define the following appropriate software metrics for the measurement and evaluation of these aspects $\{\mu_{development(MAS)}\}$.

The metrics for the ***MAS development life cycle***: *Agent system phases level* $\mu_{development(MAS)}^{phases}$: This evaluation considers the system metrics of size, structure and complexity during system development. *Agent system milestones level* $\mu_{development(MAS)}^{milestones}$: The metric evaluates MAS development with respect to a milestone. *System requirements workflow level* $\mu_{development(MAS)}^{workflow}$: The requirements implementation during the development phases in the whole system is considered.

The metrics for the ***MAS development method***: *Software agent methodology level* $\mu_{development(MAS)}^{methodology}$: The level of the development method used is quantified. *Software agent paradigm level* $\mu_{development(MAS)}^{paradigm}$: This metric evaluates the appropriateness of the chosen development paradigm. *Software agent CASE level* $\mu_{development(MAS)}^{CASE}$: This metric quantifies the tool support for the agent implementation.

The metrics for the ***MAS development management level***: *System project management level* $\mu_{development(MAS)}^{management(project)}$: The management level of the development risks and methods of the system is considered. *System configuration management level* $\mu_{development(MAS)}^{management(configuration)}$: This metrics includes the evaluation of the dynamic aspects of the system configuration. *System quality management level* $\mu_{development(MAS)}^{management(quality)}$: The quality assurance techniques related to the whole agent-based system is considered.

The agent and MAS development process require different resources such as personnel (developer, tester, administrator etc.), software resources (MAS COTS and CASE tools), and platform resources, including the hardware components. Therefore, we need measurement values with respect to the characteristics (especially the quality) of these resources. Hence, we define the following metrics which are also necessary to evaluate the MAS development process.

The metrics for the ***agent developer level***: *Agent developer skill level* $\mu_{development(Agent)}^{skill(developer)}$: This metric is related to the skills to develop and implement an software agent. *Agent developer communication level* $\mu_{development(Agent)}^{communication(developer)}$: The ability of the developer to improve his work by collaboration and cooperation is considered. *Agent developer productivity level* $\mu_{development(Agent)}^{productivity(developer)}$: This metric evaluates the quantity of work.

The metrics for the ***agent software resources level***: *Agent software paradigm level* $\mu_{development(Agent)}^{paradigm(software)}$: This metric evaluates the appropriateness of the chosen software basis and used software components for the implementation of an software agent. *Agent software performance level* $\mu_{development(Agent)}^{performance(software)}$: This metric addresses the software components and their effectiveness. *Agent software replacement level* $\mu_{development(Agent)}^{replacement(software)}$: This metric considers the effort of adaptation when using different versions of the basic software.

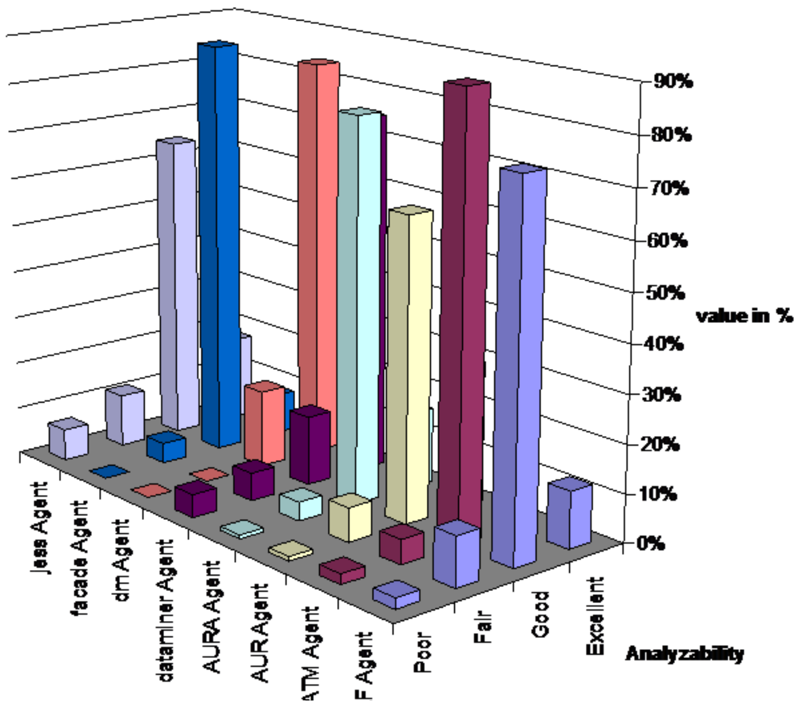


Figure 30: Example of MAS resources measurement (Agent Academy, [Wille 2005])

The metrics for the ***agent hardware resources level***: *Agent hardware reliability level* $\mu_{development(Agent)}^{reliability(hardware)}$: This metrics considers the reliability of the types of hardware required for running the software agent. *Agent hardware performance level* $\mu_{development(Agent)}^{performance(hardware)}$: This set of metrics considers the platforms used for an agent**.** *Agent hardware availability level* $\mu_{development(Agent)}^{availability(hardware)}$: The average availability of the different platforms used from a (mobile) agent is considered.

The metrics for the ***MAS developer level***: *System developer skill level* $\mu_{development(MAS)}^{skill(developer)}$: This metric is based on the agent developer skills and is extended by the (dynamic) system characteristics. *System developer communication level* $\mu_{development(MAS)}^{comunication(developer)}$: This set of metrics considers the ability of the developer(s) to improve his work by collaboration and cooperation. *System developer productivity level* $\mu^{productivity(developer)}_{development(MAS)}$: The quantity of work is considered.

The metrics for the ***MAS software resources level***: *System software paradigm level* $\mu^{paradigm(software)}_{development(MAS)}$: The appropriateness of the chosen software basis and COTS system used for the implementation of the agent-based system is evaluated. *System software performance level* $\mu^{performance(software)}_{development(MAS)}$: This metric considers the evaluation of the efficiency of the involved software base and the external components. *System software replacement level* $\mu^{replacement(software)}_{development(MAS)}$: The adaptation to the different versions of the basic software is considered.

A metrics-based analysis of different Java-based agent technologies shows the following two tables from Kernchen [Kernchen 2006] (see also [Ebert 2007]).

Table 1: Size measurements for measured AOSE technologies

System	# Classes	# Methods	LOC/Class	Methods/Class
Aglets	180	1863	67.61	10.35
JADE	487	4652	99.18	9.55
MadKit	683	5929	109.0	8.68

Considering the C&K metrics has highlighted the following characteristics between the different agent technologies as software Aglets, JADE development platform and the MadKit system.

Table 2: Chidamber and Kemerer measurements for OOSE and AOSE technologies

	Aglets	**JADE**	**MadKit**	Mean of AOSE	Std. dev. of AOSE	Mean of OOSE	Std. dev. of OOSE
DIT	0.239	0.745	0.685	0.556	0.276	0.59	0.82
NOC	0.222	0.353	0.387	0.321	0.087	0.15	0.45
WMC	10.35	9.552	8.69	9.531	0.830	8.69	7.90
CBO	5.022	6.951	5.331	5.768	1.036	3.00	4.22
RFC	25.05	15.871	21.931	20.951	4.667	14.78	16.35
LCOM	80.011	55.585	50.144	61.913	15.907	37.51	82.82

The values of the OOSE evaluation are based on the Java 1.5 measurement using the C&K metrics (see for more details [Kernchen 2006]).

The metrics for the ***MAS hardware resources level***: *System hardware reliability level* $\mu^{reliability(hardware)}_{development(MAS)}$**:** The reliability of the kinds of hardware for running the agent-based system is considered. *System hardware performance level* $\mu^{performance(hardware)}_{development(MAS)}$: This set of metrics considers the platforms used for an agent-based system. *System hardware availability* *level* $\mu_{development(MAS)}^{availability(hardware)}$: The average availability of the different platforms used with the agent-based system is considered.

4.3 Agent Technology and Measurement Levels

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, we can characterize the high level of software measurement for agent technology as following.

$$MP_{inhouse}^{capability\_oriented}: \tag{4.7}$$
$$(G_{goals\_in\_use}^{managing} \times A_{original}^{product} \times M_{measurement}^{controlling})T_{some\_measurement}^{automatic}$$
$$\to (V_{repository}^{ratio\_scale} \times U_{quasi\_standard}^{software\_unit})\,T_{some\_measurement}^{automatic}$$
$$\to E_{extension}^{formula} \times A_{controlled}^{product}$$

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.

$$MP_{inhouse}^{aspect\_oriented}: \tag{4.8}$$
$$(G_{external\_goals}^{managing} \times A_{original}^{process}$$
$$\times M_{measurement}^{improvement})T_{some\_measurement}^{semi\_automatic}, P_{measurement\_application\_staff}^{practitioner}$$
$$\to (Q_{repository}^{ordinal\_scale} \times E_{threshold}^{criteria})\,T_{some\_measurement}^{semi\_automatic}, P_{measurement\_application\_staff}^{practitioner}$$
$$\to E_{extension}^{rules\_of\_thumb} \times A_{improved}^{process}$$

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

$$MP_{product\_quality\_assurance.}^{traditional} \precsim MP_{e-Control}^{e-Measurement} \precsim MP_{agents}^{AOSE} \tag{4.9}$$

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

$$MP_{project\_controlling}^{traditional} \precsim MP_{e-Quality}^{e-Measurement} \precsim MP_{MAS\_development}^{AOSE} \tag{4.10}$$

that could be characterized as a *moderate measurement process improvement*.

5 Adaptive Measurement for Service-Oriented Systems

5.1 Characteristics of Service-Oriented Software Engineering (SOSE)

The concepts of software architecture shall be clarified by a rather classic definition cited here from [Bass 2003]:

> *"The software architecture of a program or computing system is the structure or structures of the system, which comprises software components, the externally visible properties of those components, and the relationships among them."*

In general, service-oriented architectures (SOA) can be characterized by the fact that they separate the implementation of the service from its interface. Withal, a "find, bind, and execute" paradigm enables a service's customer to query a third-party registry for an adequate service implementation. In case that the registry contains a matching service, it provides the customer with a contract and an endpoint address. Following the notes of [McGovern 2003], SOA configures its six entities, namely *service consumers, service providers, registries, contracts, proxies*, and *service leases* after all, to support the above mentioned paradigm.

The general idea of SOSE could be characterized in the model-based description of the OASIS standardization committee shown in the following figure [MacKenzie 2006].

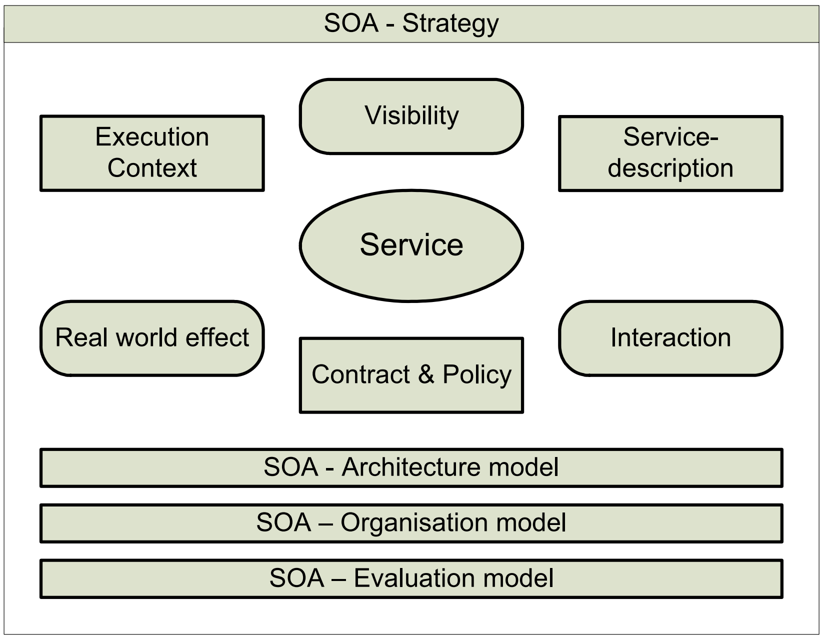


Figure 31: SOA reference model by OASIS

The general involved technology in Web services is given in the figure 32 below starting at the SOAP level. Web services are network-based applications that use the WSDL protocol (Web Service Description Language) to describe the functions they offer on the Internet, XML documents (eXtensible Markup Language) to exchange information, and the SOAP protocol (Simple Object Access Protocol) for calling remote methods and transferring data. The data and function calls that are packaged into XML documents are typically transferred using the http protocol which means communication can also take place across firewalls. It is this property in particular that opens up the possibility of developing genuine Business to Business (B2B) applications. UDDI directory services (Universal Description, Discovery, and Integration) are used to localize the Web services provided on the Internet.

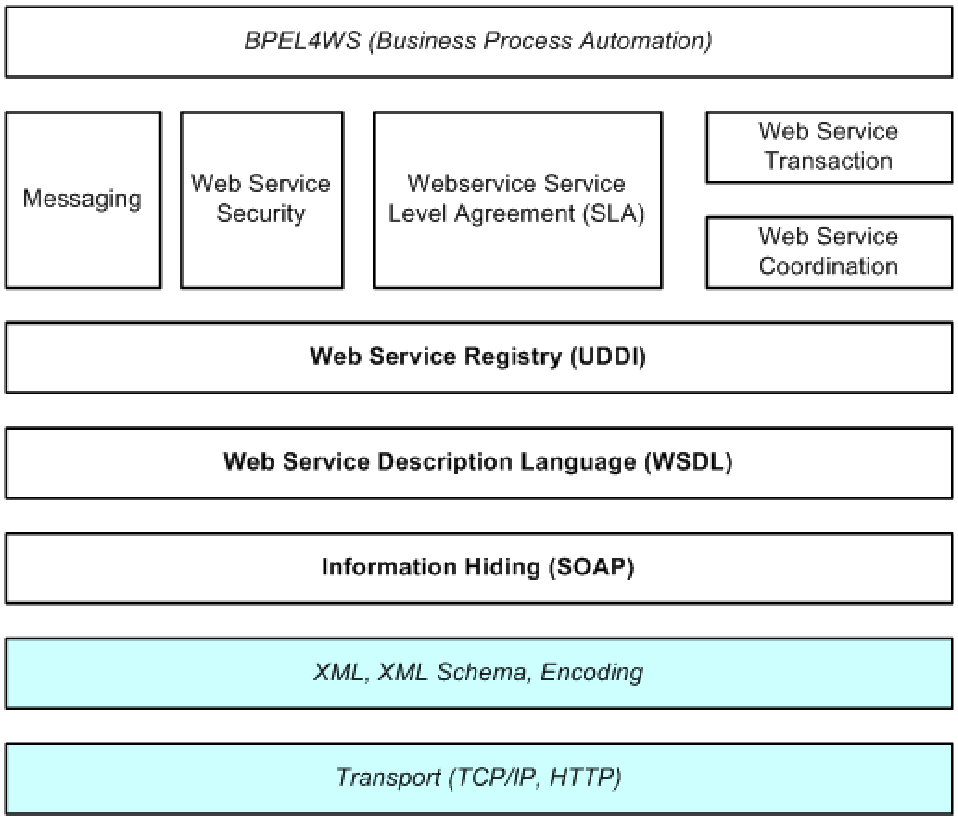

BPEL4WS (Business Process Automation)
Messaging
Web Service Security
Webservice Service Level Agreement (SLA)
Web Service Transaction
Web Service Coordination
Web Service Registry (UDDI)
Web Service Description Language (WSDL)
Information Hiding (SOAP)
XML, XML Schema, Encoding
Transport (TCP/IP, HTTP)

Figure 32: Example of SOA architectural basics

[Hanson 2006] describes different levels of granularity. He points out that the level of granularity generally depends upon the purpose of the software entity. The different levels are shown in figure 32. The level of granularity for composite services should be coarser than the level of granularity for basic services, objects or components. Figure 33 shows furthermore the kinds of development support. Objects and components support more than the development of single applications (application centric). Basic services and composite services support the development of IT-architectures (architecture centric) for the entire enterprise. That means with the application of services, development of individual applications does not stand in the foreground any more.

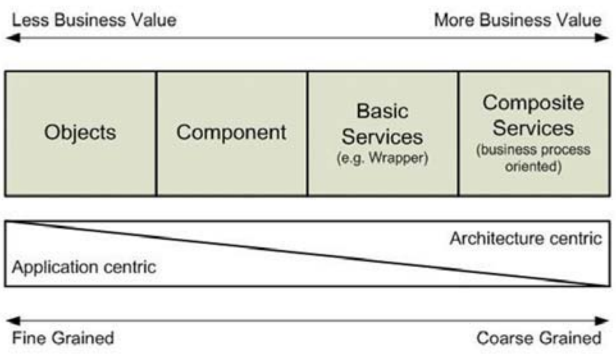


Figure 33: Degrees of service granularities by [Hanson 2006]

A typical SOA in an industrial environment is given in the following figure existing in the telecommunication area (*eLoC* means effective lines of code).

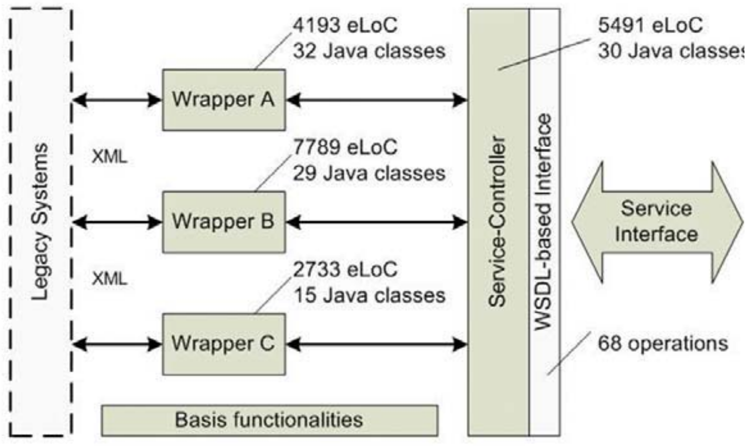


Figure 34: Architecture of an industrial SOA service by [Schmietendorf 2007b]

The service development must consider the existing SOA-infrastructure from the customer side. That means that the service should be usable within the established runtime-environment.

The right granularity of corresponding service offers is crucial for the successful implementation of a SOA. Object-oriented, component-based and service-oriented software engineering paradigms have many resembling features – modularity, encapsulation of functionality and data, separation of interface and implementation, and so on. Therefore, it could be possible to derive experiences from this field. [Griffel 1998] proposes the following set of metrics to measure the granularity indirectly:

- Size of the service interface (operations, parameter, …)
- Share of the service of the complete application (like supported business processes)
- Size of the effective source code (lines of code, number of classes, …)

A first approach for an evaluation model follows the well known GQM (goal question metric) paradigm and leads to the granularity identification by the use of metrics. Therefore an assessment for the granularity behavior of a service offering should provide answers to the following questions:

How much should be the size of the service interface?

(Number of provided operations, Number of contained data types for each operation, Kinds of used data types)

How much business functionality is considered by the service?

(Information about the successful application, Number of encapsulated legacy systems, Difference between basic and composite services)

Is a "white box" or/and "black box" view supported?

(Information about the used implementation technology, Structure metrics for the service implementation, Overview of the chosen implementation architecture)

5.2 SOSE addressed Measurement Descriptions

Typical measurements in the context of a SOA scorecard could refer to the following areas [Schmietendorf 2007a]:

- SOA – Business Measurements: *Market penetration, time to market, customer satisfaction, turnover increase* etc.
- SOA – Process Measurements: *Process terms, process mistakes, number of process-referential events* etc.
- SOA – Financial measurements: *Return of Investment, cost savings, project costs* etc.
- SOA – Usage measurements: *Number of used service offerings, number of service customers* etc.
- SOA – Performance measurements: *Performance of basic and orchestrated Services and availability* etc.
- SOA – IT efficiency measurements: *Productivity, development time, quality behavior* etc.
- SOA – Optimization measurements: *Number of services in conception, development and production* etc.
- SOA – Governance measurements: *Standard conformity, potential exceptions* etc.

A general characterization of service measurement could be explained in the following manner considering the autonomous behaviour of services themselves.

$$\begin{aligned}
&MP_{services}^{SOSE}: \qquad (5.1)\\
&(G_{QoS}^{managing} \times A_{original}^{product} \times M_{measurement}^{controlling})T_{some\_measurement}^{automatic}\\
&\rightarrow (V_{data\_basis}^{ratio\_scale} \times U_{quasi\_standard}^{software\_unit})\,T_{some\_measurement}^{automatic}\\
&\rightarrow E_{extension}^{service\_level} \times A_{controlled}^{product}
\end{aligned}$$

Especially, the measurement methods $\mu_{service}$ can be summarized as (based on [Rud 2006a] and [Rud 2006c]; see also [Cardoso 2006], [Hiekel 2007], [Kalepu 2003], [Perepletchikov 2008], [Thielen 2004] and [Yu 2007])

$$\begin{aligned}
&measurement \in \{\, \mu_{CY}^{complexity}, \mu_{NSIC}^{complexity}, \mu_{SIY}^{complexity}, \qquad (5.2)\\
&\mu_{CSES}^{criticality}, \mu_{ADS}^{criticality}, \mu_{AIS}^{criticality}, \mu_{ACS}^{criticality}, \mu_{RC}^{criticality},\\
&\mu_{DAS}^{granularity}, \mu_{DCCS}^{granularity}, \mu_{CDOS}^{granularity}, \mu_{DGOS}^{granularity} \}
\end{aligned}$$

The evaluation of a service as *product* is based on the different metrics of complexity, criticality and granularity. The ***product/service metrics of complexity*** are: Cohesion of the system $\mu_{CY}^{complexity}$, number of services involved in the compound service $\mu_{NSIC}^{complexity}$ and service independence in the system $\mu_{SIY}^{complexity}$.

The ***product/service metrics of criticality*** are: Count of semantic equivalents of the service $\mu_{CSES}^{criticality}$, absolute importance of the service $\mu_{AIS}^{criticality}$, absolute dependence of the service $\mu_{ADS}^{criticality}$, absolute criticality of the service $\mu_{ACS}^{criticality}$ and overall reliability of the compound service $\mu_{RC}^{criticality}$.

The ***product/service metrics of granularity*** are: Domains affected by the service $\mu_{DAS}^{granularity}$, domains completely covered by the service $\mu_{DCCS}^{granularity}$, context-dependence of operations of the service $\mu_{CDOS}^{granularity}$ and data-oriented granularity of operations of the service $\mu_{DGOS}^{granularity}$.

The following example of Web service measurement is based on the architecture shown in figure 34 [Schmietendorf 2007b]. Under the consideration of the messages within the SOAP-header, the number for the transmitted parameters increases. The sum of all transmitted parameters per operation (request and response) is shown in figure 35. All request operations contain the same kind of license information.

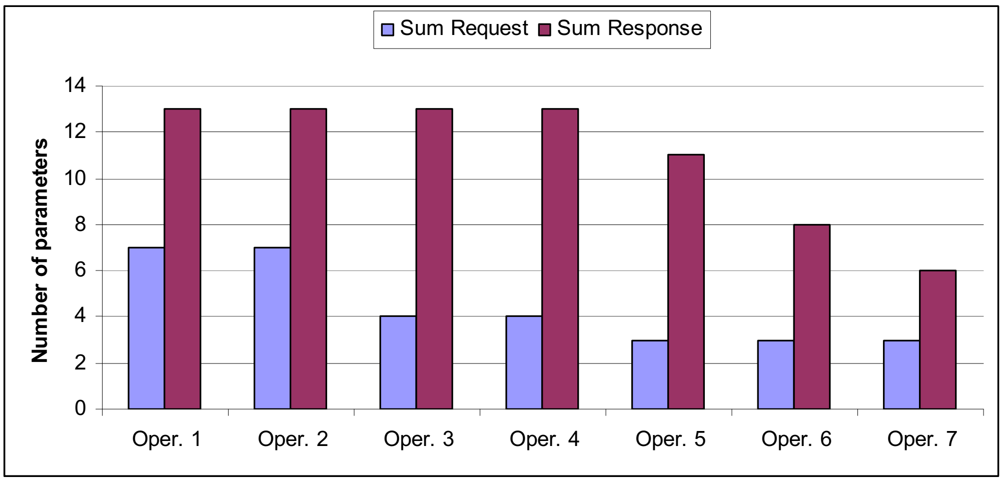


Figure 35: Characteristics of an industrial SOA service by [Schmietendorf 2007b]

A general characterization of measurement of the service development could be explained in the following manner.

$$MP_{service\_development}^{SOSE}: \tag{5.3}$$

$$(G_{external\_goals}^{managing} \times A_{original}^{process} \times M_{measurement}^{assessment})T_{some\_measurement}^{semi\_automatic}, P_{measurement\_application\_staff}^{practitioner}$$

$$\rightarrow (Q_{repository}^{ordinal\_scale} \times E_{threshold}^{criteria})\ T_{some\_measurement}^{semi\_automatic}, P_{measurement\_application\_staff}^{practitioner}$$

$$\rightarrow E_{extension}^{methodology} \times A_{managed}^{process}$$

Especially, the measurement methods $\mu_{dev.(service)}$ can be summarized as (based on [Rud 2007a] and [Rud 2007b])

$$measurement \in \{\ \mu_{CY}^{integrity},\ \mu_{QMM}^{maturity}\ \} \tag{5.4}$$

The evaluation of a service development as *process* is based on the different metrics of integrity and process maturity. The ***process metrics of integrity*** is: Integrated process quality of SOA adoption $\mu_{CY}^{integrity}$.

The ***process metrics of maturity*** are: Quality of the maturity model $\mu_{QMM}^{maturity}$ where QMM could be based on *SOA-MM*$_{Linthicum}$, *SOA-MM*$_{Sonic}$, *SOA-MM*$_{IBM}$, *SOA-MM*$_{Oracle}$, *SOA-MM*$_{BEA}$, *SOA-MM*$_{EDS}$ etc. (see [Rud 2007b] for more details and model ranking).

Finally, a general characterization of measurement service in use could be explained in the following manner.

$$\mathbf{MP}_{service\_application}^{SOSE} : \tag{5.5}$$

$$(G_{SLA}^{managing} \times A_{original}^{product \wedge resources} \times M_{measurement}^{controlling})T_{some\_measurement}^{automatic}$$

$$\rightarrow (V_{data\_basis}^{ratio\_scale} \times U_{quasi\_standard}^{economical\_unit})\ T_{some\_measurement}^{automatic}$$

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

Especially, the measurement methods $\mu_{appl.(service)}$ can be summarized as (based on [Rud 2007a])

$$measurement \in \{ \mu_{CVS}^{versioning}, \mu_{ACSVY}^{versioning}, \mu_{ALTVS}^{versioning}, \mu_{ALTSVY}^{versioning}, \mu_{MCFS}^{versioning}, \tag{5.6}$$
$$\mu_{MCFY}^{versioning}, \mu_{SLACS}^{reliability}, \mu_{SLAVDS}^{reliability}, \mu_{FRO}^{reliability}, \mu_{FRY}^{reliability},$$
$$\mu_{ANBPY}^{performance}, \mu_{BPCY}^{performance} \}$$

The evaluation of a service application is based on the different metrics of versioning, reliability and performance. Note that the service application is a non trivial process of orchestration, optimization and autonomous application. The ***service application metrics of versioning*** are: Count of simultaneous versions of the service $\mu_{CVS}^{versioning}$, average count of services' versions in the system $\mu_{ACSVY}^{versioning}$, average life time of versions of the service $\mu_{ALTVS}^{versioning}$, average life time of services' versions in the system $\mu_{ALTSVY}^{versioning}$, metadata change frequency of the service $\mu_{MCFS}^{versioning}$ and overall metadata change frequency in the system $\mu_{MCFY}^{versioning}$.

The ***service application metrics of reliability*** are: SLA compliance of the service $\mu_{SLACS}^{reliability}$, SLA violation danger of the service $\mu_{SLAVDS}^{reliability}$, fault rate of the operation $\mu_{FRO}^{reliability}$ and overall fault rate in the system $\mu_{FRY}^{reliability}$.

The ***service application metrics of performance*** are: Average number of business processes in the system $\mu_{ANBPY}^{performance}$ and business process capacity of the system $\mu_{BPCY}^{performance}$.

A simple example of the analysis of service-based resources for chosen Web services shows the following situation of the used SOA technologies [Schmietendorf 2002].

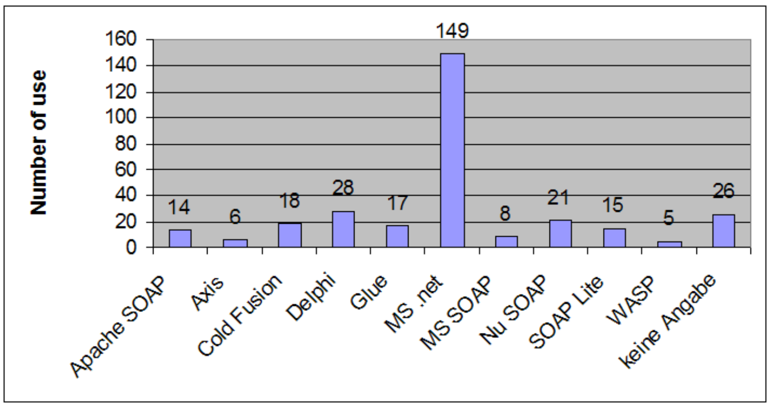
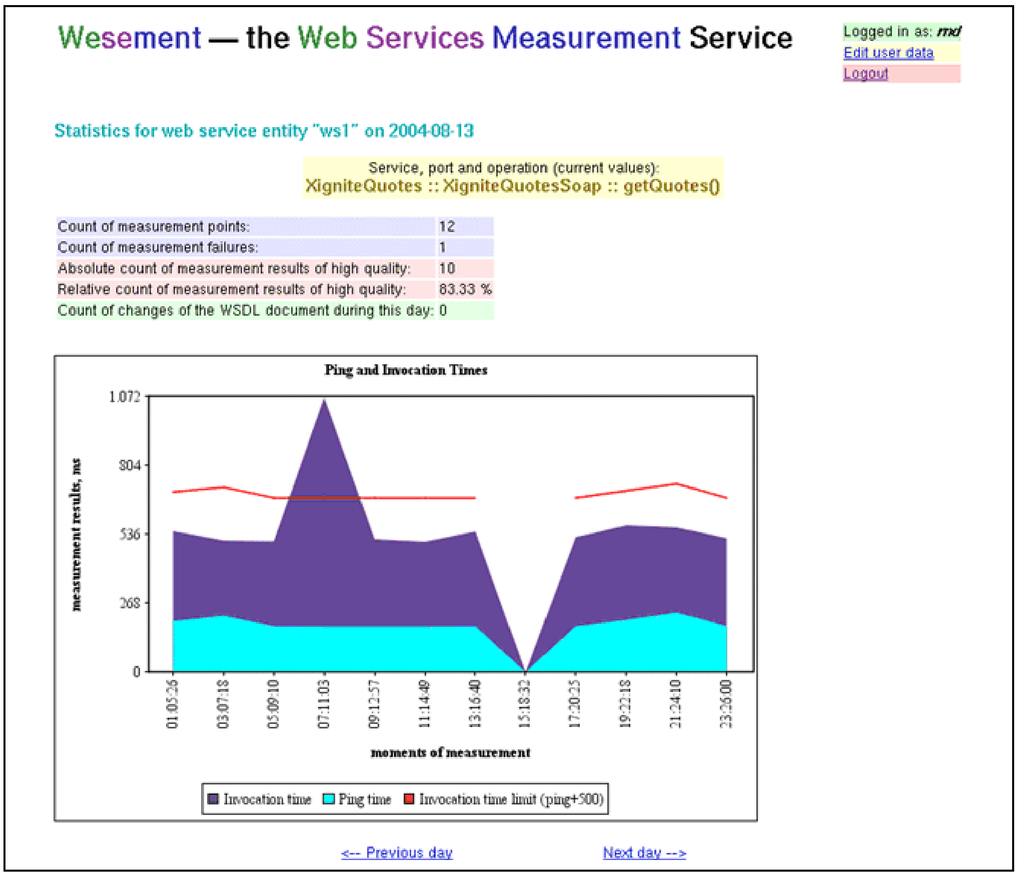


Figure 36: Contribution of technologies for existing Web services

Another example analyses Web services considering their availability, performance and quality of description was implemented as a service itself (available since 2006, [Rud 2006a]).


Wesement — the Web Services Measurement Service

Logged in as: *md*
Edit user data
Logout

Statistics for web service entity "ws1" on 2004-08-13

Service, port and operation (current values):
XigniteQuotes :: XigniteQuotesSoap :: getQuotes()

Count of measurement points:	12
Count of measurement failures:	1
Absolute count of measurement results of high quality:	10
Relative count of measurement results of high quality:	83.33 %
Count of changes of the WSDL document during this day:	0

<-- Previous day
Next day -->

Figure 37: The Wesement service of continued Web service evaluation

The BPEL engine of Rud could be considered as a kernel approach of service measurement described in [Rud 2006b] as *BPELmeter*.

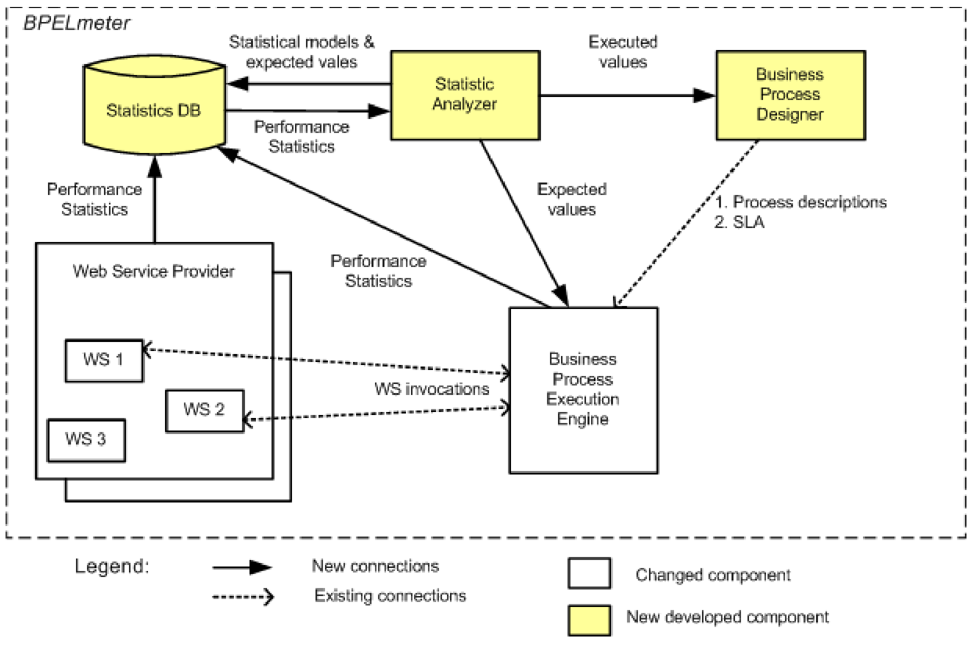


Figure 38: Infrastructure of the BPELmeter

Finally, a measurement service based on the shown architecture in figure 39 was implemented [Ebert 2007]. It has the possibility to measure the availability, the performance, the functionality and the complexity of a specific Web service from the users (or better integrators) point of view.

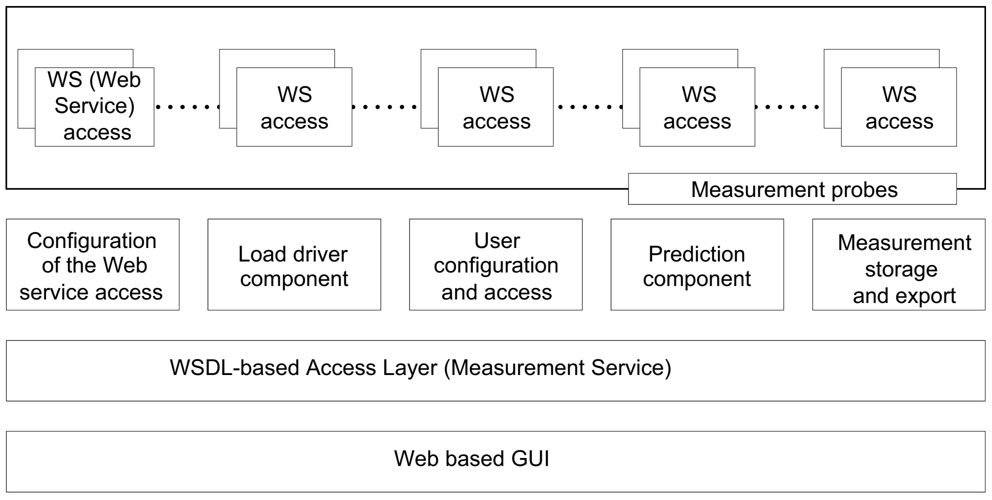

WS (Web Service) access
WS access
WS access
WS access
WS access
Measurement probes
Configuration of the Web service access
Load driver component
User configuration and access
Prediction component
Measurement storage and export
WSDL-based Access Layer (Measurement Service)
Web based GUI

Figure 39: Architecture of a measurement service

It provides a simple graphical control interface. Provides the configuration of the measurements time interval and measurement goals and generates simple graphical reports too.

5.3 SOSE intended Measurement Levels

The measurement levels in SOSE will be characterized as the three areas above. The first described level is addressed to the service measurement and could be explained in the following manner considering the autonomous behaviour of services themselves (like agents).

$$MP^{aspect\_oriented}_{inhouse}: \tag{5.7}$$
$$(G^{managing}_{internal\_goals} \times A^{product}_{original} \times M^{controlling}_{measurement})T^{automatic}_{some\_measurement}$$
$$\rightarrow (V^{ratio\_scale}_{data\_basis} \times U^{software\_unit}_{quasi\_standard})\ T^{automatic}_{some\_measurement}$$
$$\rightarrow\ E^{formula}_{extension} \times A^{product}_{controlled}$$

Note that the service measurement includes its application from the beginning and leads to a short cycle of development and application.

The second area of SOSE considers the measurement of the service development and could be explained in the measurement level as following.

$$MP^{aspect\_oriented}_{inhouse}: \tag{5.8}$$
$$(G^{managing}_{external\_goals} \times A^{process}_{original} \times M^{assessment}_{measurement})T^{semi\_automatic}_{some\_measurement},\ P^{practitioner}_{measurement\_application\_staff}$$
$$\rightarrow (Q^{ordinal\_scale}_{repository} \times E^{criteria}_{threshold})\ T^{semi\_automatic}_{some\_measurement},\ P^{practitioner}_{measurement\_application\_staff}$$
$$\rightarrow\ E^{rules\_of\_thumb}_{extension} \times A^{process}_{managed}$$

Finally, the third area is addressed to the measured service application and achieves the following measurement level usually.

$$MP^{capability\_oriented}_{outsourced}: \tag{5.9}$$
$$(G^{managing}_{goals\_in\_use} \times A^{product \wedge resources}_{original} \times M^{controlling}_{measurement})T^{automatic}_{some\_measurement}$$
$$\rightarrow (V^{ratio\_scale}_{data\_basis} \times U^{economical\_unit}_{quasi\_standard})\ T^{automatic}_{some\_measurement}$$
$$\rightarrow\ E^{formula}_{extension} \times A^{product}_{controlled}$$

The difference of the in-house and outsourced characteristic between service measurement and service application measurement is reasoned in the external view of the choreographic and orchestration aspects.

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

$$MP^{traditional}_{product\_quality\_assurance.} \prec MP^{SOSE}_{services} \prec MP^{AOSE}_{agents} \tag{5.10}$$

and considering the more complex training phase in service development as *moderate measurement process improvement* as

$$MP^{traditional}_{project\_controlling} \prec MP^{AOSE}_{MAS\_development} \prec MP^{SOSE}_{service\_development}. \tag{5.11}$$

Furthermore, as including the (measured) resources in service application as a *essential measurement process improvement*

$$MP^{AOSE}_{MAS} \prec MP^{SOSE}_{service\_application}. \tag{5.12}$$

6 Measurement Infrastructures as Proactive Measurement

6.1 Intention and Examples of Measurement Infrastructures

The importance of software measurement during the software development process is generally accepted, nowadays. Unfortunately, in practice common software measurement tools find small acceptance due to their high costs, inflexible structures, and therewith unclear cost/benefit ratio. On this account this section introduces a framework creating a measurement infrastructure by means of a service-oriented architecture. For this approach the ISO/IEC 15939 standard has been proved to be meaningful [ISO 15939]. By using meta-models and ontologies, related services can be categorized and/or classified. Moreover, services can be bound flexibly with the aid of configuration defaults being referenced by the meta-model. Furthermore, we present a web-service based ontology aligned towards object-oriented metrics as an example for a service-oriented infrastructure component.

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 realise the defined processes and activities [Dumke 2005b].

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 40.

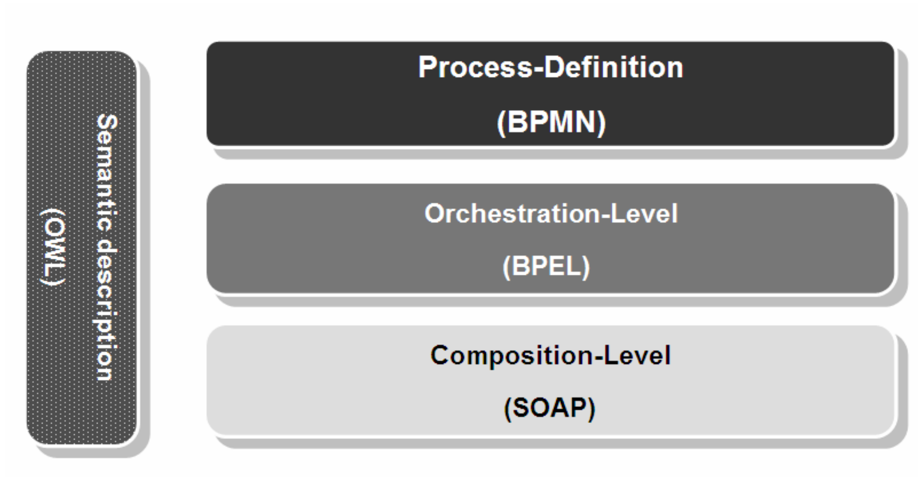


Figure 40: Level-based infrastructure composition

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, we apply the Business Process Modelling Notation (BPMN) [BPMN 2005]. 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, we use the BPMN to produce the business process diagram (figure 41) on the basis of the ISO/IEC 15939 process model.

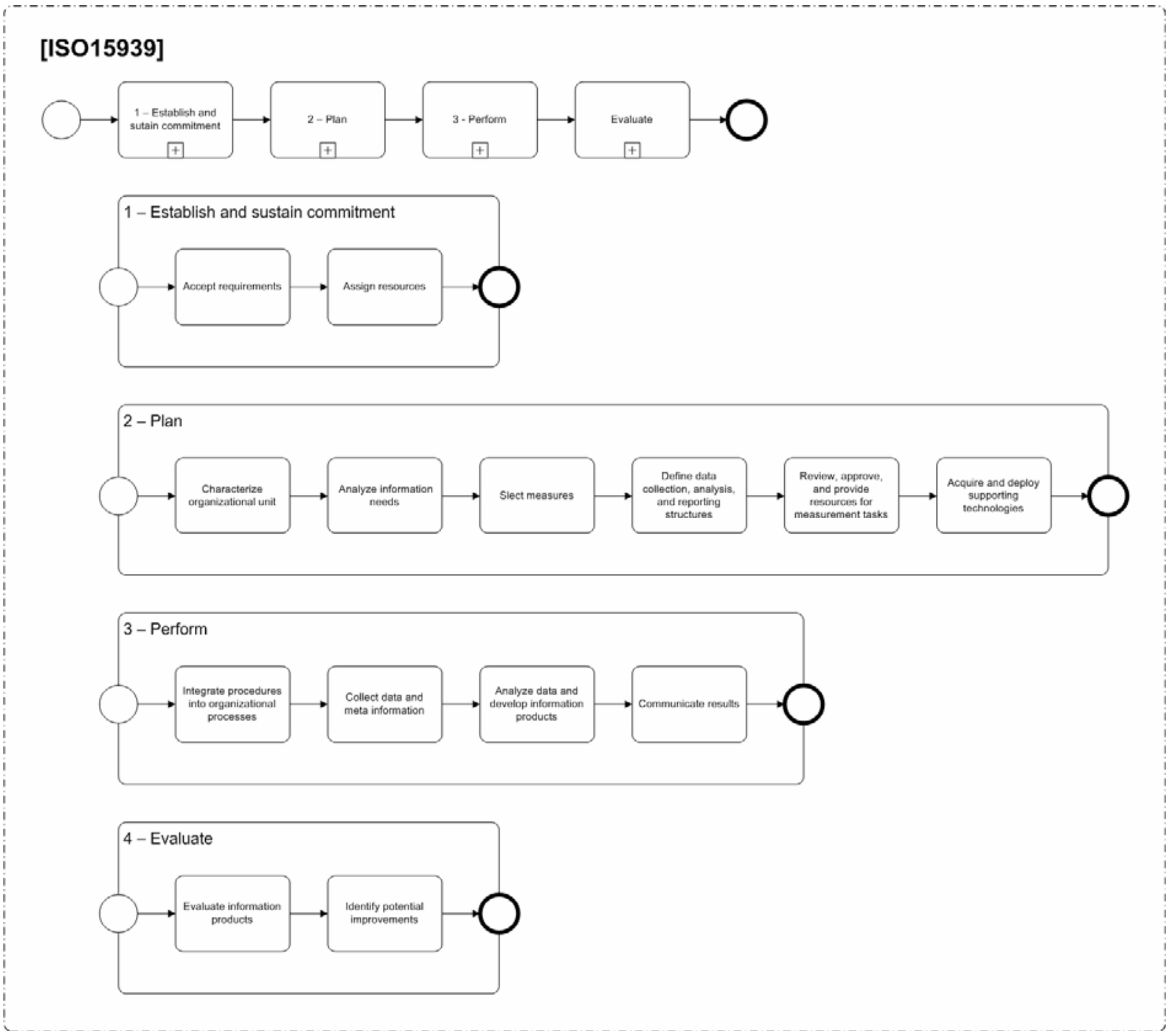

[ISO15939]
1 – Establish and sutain commitment
2 – Plan
3 - Perform
Evaluate
1 – Establish and sustain commitment
Accept requirements
Assign resources
2 – Plan
Characterize organizational unit
Analyze information needs
Slect measures
Define data collection, analysis, and reporting structures
Review, approve, and provide resources for measurement tasks
Acquire and deploy supporting technologies
3 – Perform
Integrate procedures into organizational processes
Collect data and meta information
Analyze data and develop information products
Communicate results
4 – Evaluate
Evaluate information products
Identify potential improvements

Figure 41: Business process diagram for the ISO 15393 measurement standard

By using such representation one can derive the Business Process Execution Language (BPEL) by using BPEL4WS (see also [BPMN 2005]).

This Business Process Execution Language leads us to a key technology for realising service-oriented architectures: the web service orchestration [Erl 2005]. As shown in figure 42 the orchestration process is used to build up new web services out of existing web services in a hierarchical manner.

In the service-oriented infrastructure the web service orchestration is used for the composition of the ISO/IEC 15939 process out of the four sub-processes (“establish”, “plan”, “perform”, and “evaluate”) in the so called Orchestration Process (OP).

In this way the result of the orchestration process is four equitable sub-processes. One has to recognise that collaboration of the four sub-processes has to be performed in a peer-to-peer manner. Therefore, the different sub-processes are divided into four choreography processes (CP) [Erl 2005]. The difference between orchestration and choreography means that orchestration refers to an executable process and choreography traces the message sequence between different sources [Peltz 2003].

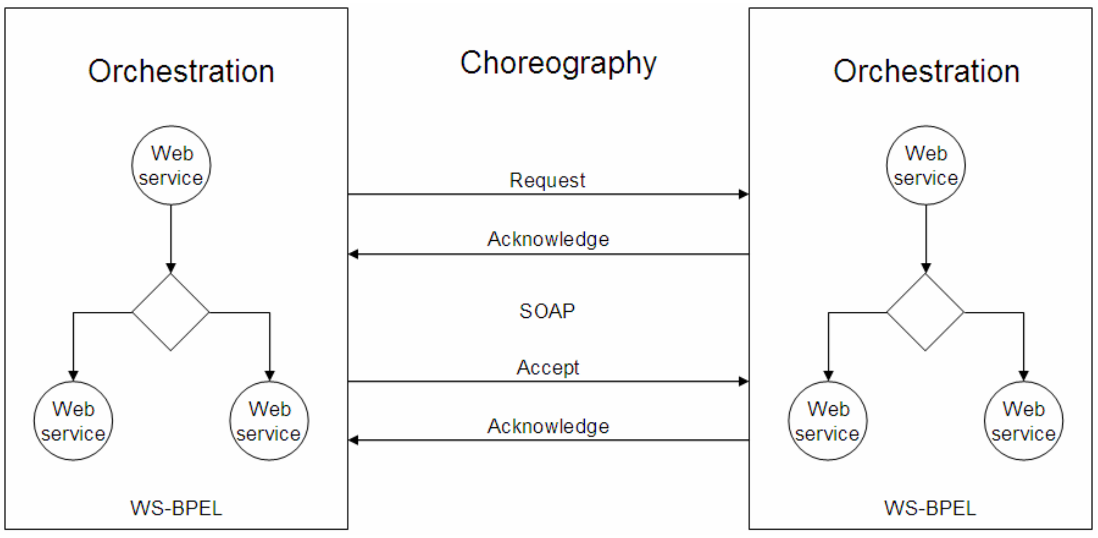


Figure 42: Orchestration versus choreography [Peltz 2003]

Because of the fact that the choreography process implies the orchestration process, we define the orchestration process as the level 1 process and the choreography process as the level 2 process (see figure 43).

By taking a closer look to the four sub-processes one has to identify which of the sub-processes can be executed by the presented infrastructure or which sub-processes can merely be supported by our infrastructure. In doing so, one has to ascertain that the "plan" sub-process will be the key process in a service-oriented measurement infrastructure. Because of that figure 43 describes a set of required components with a focus on the "plan" sub-process. The colour of the component shows to which composition level (see figure 40) the distinct component belongs.

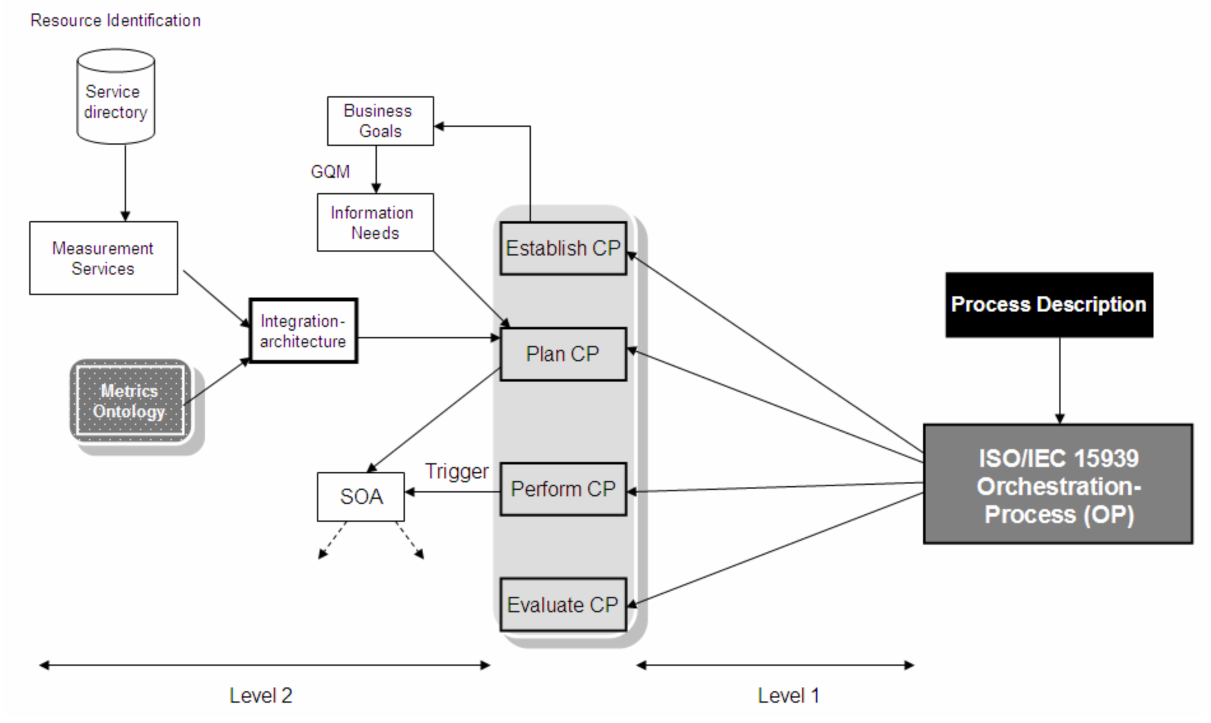


Figure 43: Fragment of the proposed infrastructure

The semantic description as shown in figure 40 is realised within a metrics ontology (see figure 43). In history ontologies possessed the capability to retain this semantic knowledge in a machine-accessible manner. Therefore, we use the ontology approach for a cataloging web system [Martin 2003] to create our own ontology for a subset of metrics (object-oriented metrics). Thereby, the ISO/IEC 9126 product quality standard is used to categorise the metrics. In general the ontology is used to connect an information need with a certain metric.

That means that all metrics which are calculated by an included measurement service must be described within this ontology. At the moment we restrict our ontology to object-oriented metrics. The ontology scheme is illustrated in figure 44 by using the Unified Modelling Language (UML) [Wang 2001].

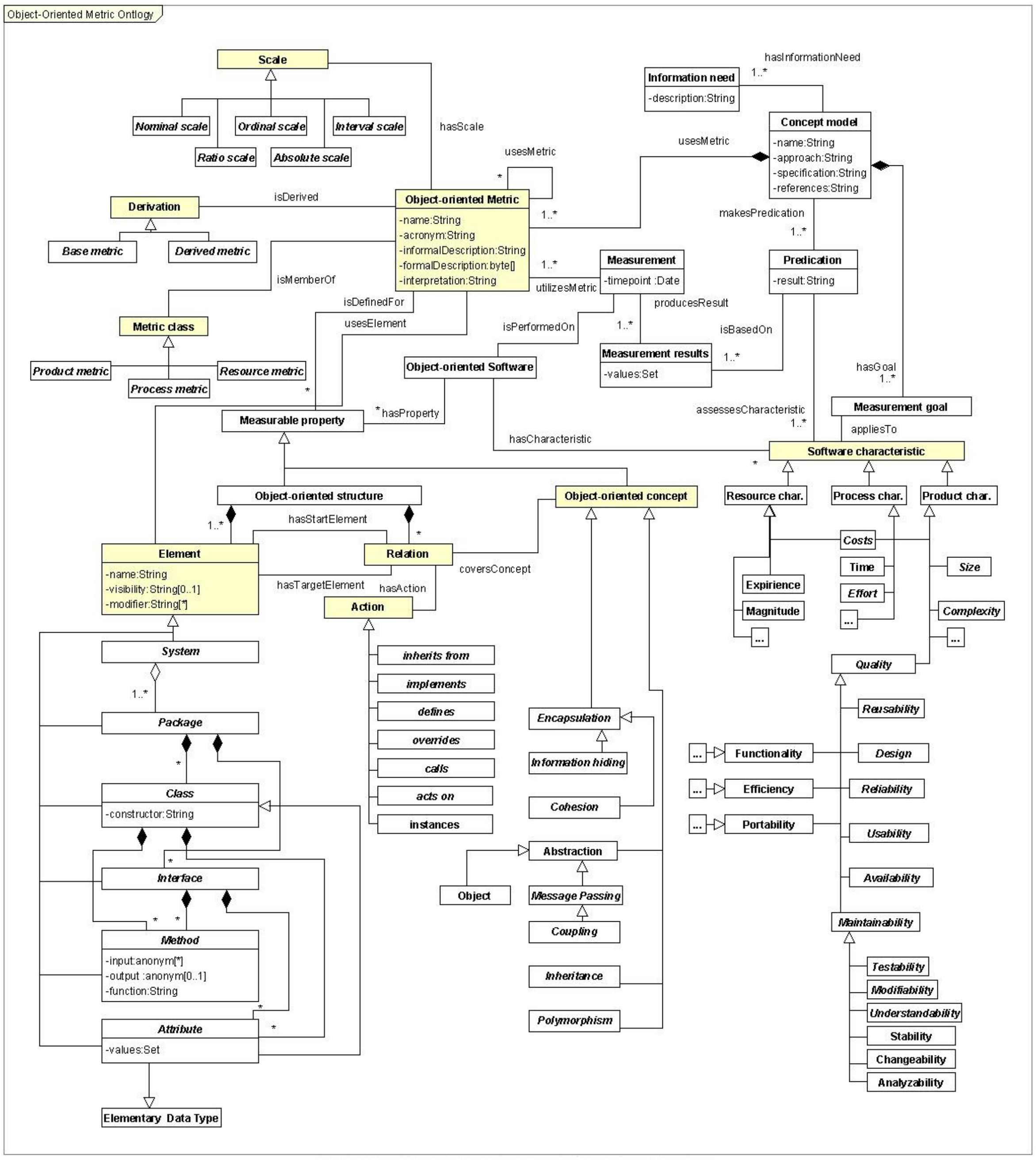


Figure 44: Object-oriented metrics ontology

In our context, the essential intentions and motivations for ***measurement infrastructures***[3] can be described as following:

- Measurement infrastructures includes some of the main characteristics of the described *software technology paradigms* in this technical report:
    - This infrastructures could be considered as a kind of e-Measurement as ***e-Services*** especially,
    - The requested autonomous characteristic of the built measurement services involves any basics of ***agent-based*** and ***self-managed systems,***
    - Combining the characteristics above the measurement infrastructures represents the ***(Web) service paradigm*** obviously.
- Measurement infrastructures themselves are based on the *Semantic Web* architectural basis such as
    - ***Measurement ontology*** for different software paradigms (OOSE, CBSE, SOSE etc.),
    - ***Proactivity in measurements*** depending on determination of ad hoc requirements or critical situations and their proactive evaluation and decision,
    - ***Evolutionarity of measurement*** by different technologies for measurement- and experience-based service extension and adaptation,
- Measurement infrastructures should be oriented to a higher *measurement level* considering the higher complexity and criticality of the software systems. That means
    - Higher measurement level as ***capability-oriented measurement*** for product and process areas and involvements based on *essential* and *remarkable measurement process improvements* mainly,
    - ***Measurement integration*** in order to achieve a measurement application as improvement and controlling in the most relevant cases (outsourced measurement should have the same level like inhouse measurement in this context).

The typical measurement level in measurement infrastructure solutions should be achieved by the following characteristics.

$$MP^{infrastructure}_{measurement\_service}: \tag{6.1}$$

$$(G^{managing}_{goals\_in\_use} \times A^{product \wedge resources}_{original} \times M^{controlling}_{measurement})T^{semi\_automatic}_{some\_measurement}$$

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

$$\rightarrow E^{formula}_{extension} \times A^{product}_{controlled}.$$

That should be characterized as $MP^{capability\_oriented}_{inhouse}$ and should involve the $P^{practitioner}_{measurement\_application\_staff}$ for the first development phase only.

[3] Our use of the term infrastructure means a technology-based integration of Web-based systems, agent and service technologies themselves.

On the other hand, the measurement infrastructure development could be characterized as following

$$MP^{infrastructure}_{meas.\_service\_development}: \tag{6.2}$$

$$(G^{managing}_{external\_goals} \times A^{process}_{original}$$

$$\times M^{assessment}_{measurement})T^{semi\_automatic}_{some\_measurement}, P^{practitioner}_{measurement\_application\_staff}$$

$$\rightarrow (Q^{ordinal\_scale}_{repository} \times E^{criteria}_{threshold})\, T^{semi\_automatic}_{some\_measurement}, P^{practitioner}_{measurement\_application\_staff}$$

$$\rightarrow E^{rules\_of\_thumb}_{extension} \times A^{process}_{managed}.$$

In order to fulfil the criteria above we want to develop a measurement service solution that should be based on the development characteristics of (6.2) at the beginning and should achieve the application situation described in (6.1). This requires a high flexibility and possibility of aggregation between existing measurement tools and services.

The following example includes an analysis about the current situation from the tool vendor point of view. One of the results is shown in the following figure considering the openness of the measurement tools [Schmietendorf 2007b].

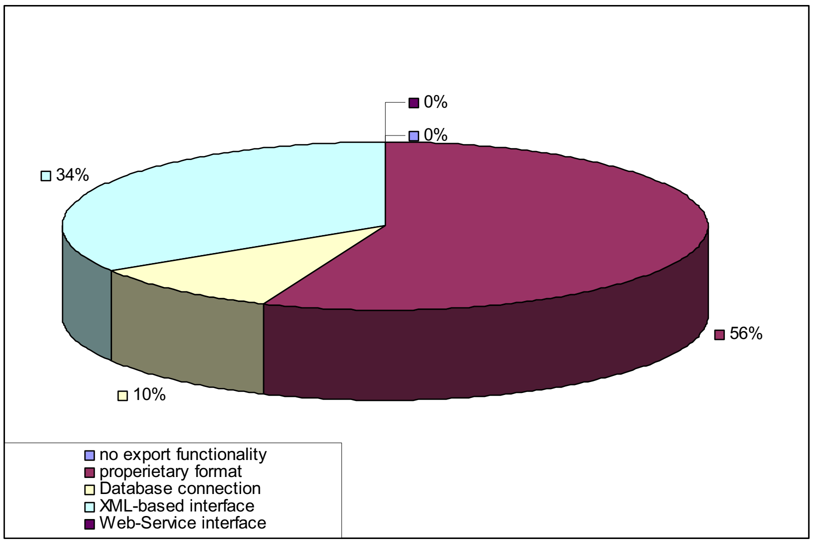


Figure 45: Analysis of service orientation of chosen measurement tool vendors

Furthermore, we need semantic descriptions for all aspects of measurement process components and involvements in order to implement first solutions in the manner of proactivity and self-managing.

The following example shows an agent-based modelling of the detailed operationalities specified by the ISO 15939 standard itself [Dumke 2005b].

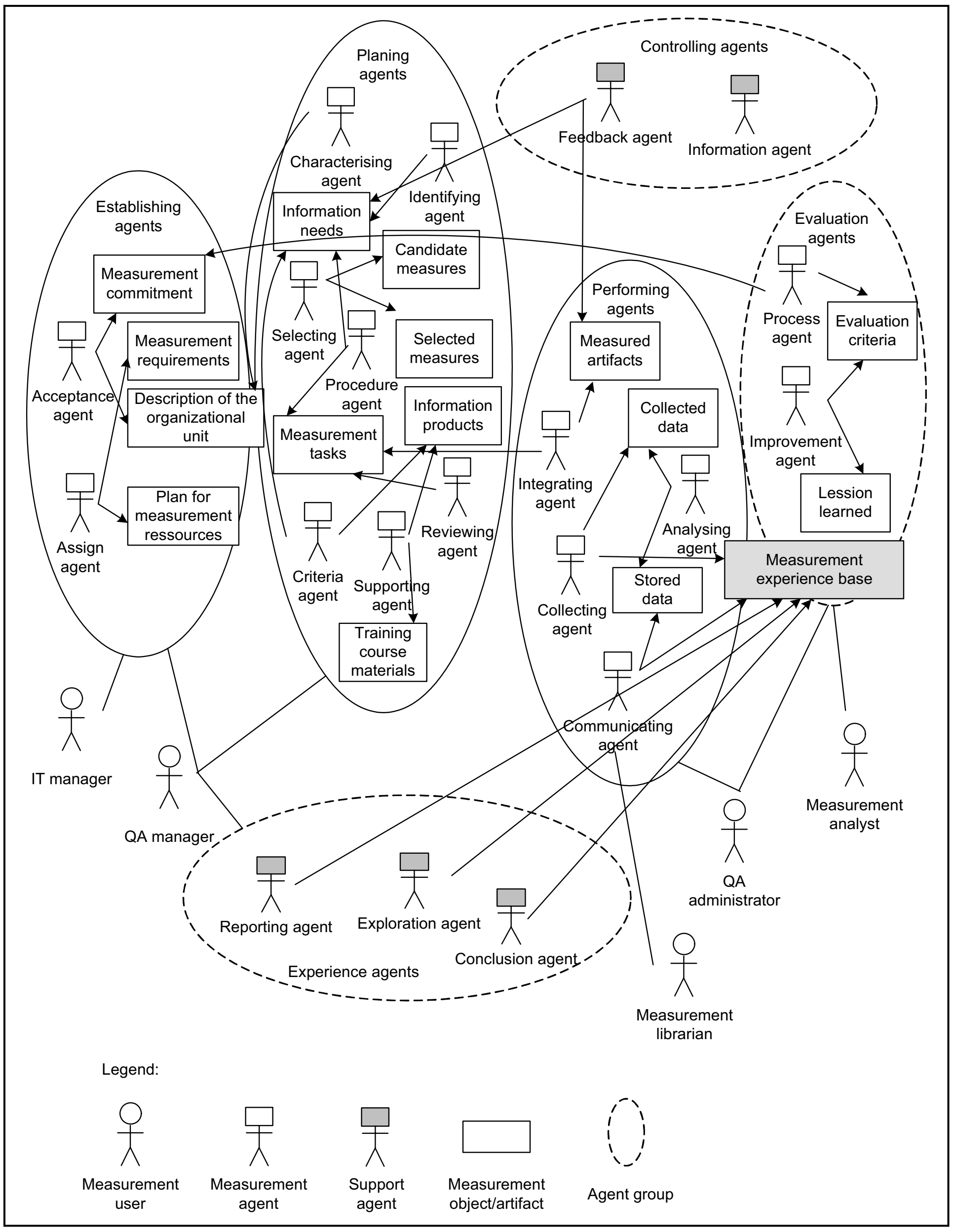


Figure 46: Ontology and agent-based measurement infrastructure for ISO 15939

The kernel process of this self-managed open system including proactivity and adaptation as ubiquitous solution is based on the knowledge-based dynamic quality assurance described in the next section.

6.2 The QuaD^2 Approach of Dynamic Quality Assurance

The QuaD^2 was published in any conferences ([Kunz 2006a], [Kunz 2006b], [Kunz 2008] and [Mencke 2008]) and led to many feedbacks for improvement and completion.

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 a thorough consideration of empirical data. Either only functional requirements or single quality attributes are taken into consideration. In contrast to existing approaches the QuaD^2 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 are considered as well as empirical evaluations. Furthermore different types of quality evaluations 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 a lot of different fields of application, e.g. service-oriented architectures or enterprise application integration. In general the sub processes of this empirical-based assembly process are the initialization, the feasibility check (checking the functional coverage), 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. 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 serve as descriptions for the functionality of the aspired distributed system. Example for such processes can be ISO/IEC 15939 [ISO 15939] for the software measurement process or didactical approaches [Mencke, 2008]. Technological realization may vary, too. That can result in UML, BPMN [BPMN 2005], ontologism [Mencke 2007], 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. 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 [Basili 1994], [Basili 1999]. 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[4].

[4] See the publications above in order to understand the special meaning of the used symbols.

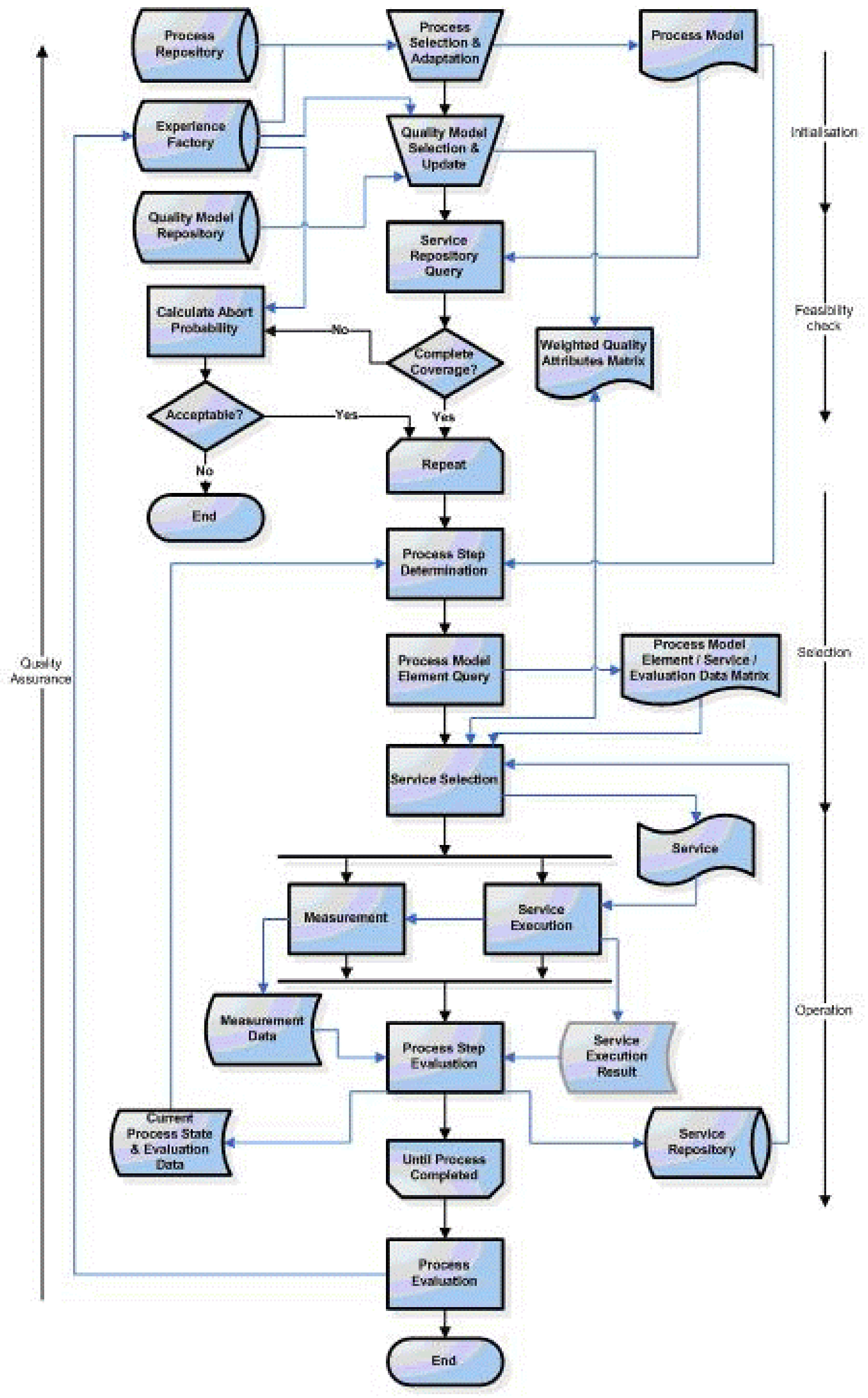


Figure 47: Quad^2 framework workflow

The focus on quality is a thorough 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 in more detailed in subsequent sections. The derived results are directly used for optimization purposes.

Initialisation 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 is essentially based 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 it means an *essential measurement improvement* combining the artefact descriptions in (6.1) and (6.2) as

$$A^{\mathit{product} \wedge \mathit{resources}}_{\mathit{original}} \cup A^{\mathit{processs}}_{\mathit{original}} = A^{\mathit{product} \wedge \mathit{resources} \wedge \mathit{process}}_{\mathit{original}} \tag{6.3}$$

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 we can establish the experience basis in the measurement process as *essential measurement improvement* (considering (6.1) and (6.2) again)

$$E^{\mathit{criteria}}_{\mathit{threshold}} \cup E^{\mathit{formula}}_{\mathit{extension}} \rightarrow E^{\mathit{formula}}_{\mathit{repository}} \,. \tag{6.4}$$

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 the last/fourth step can start. The involved transformation of measurement results could be characterized as *remarkable measurement improvement* as

$$(Q^{\mathit{ordinal\_scale}}_{\mathit{repository}} \times E^{\mathit{criteria}}_{\mathit{threshold}}) \rightarrow (V^{\mathit{ratio\_scale}}_{\mathit{data\_basis}} \times U^{\mathit{software\_unit}}_{\mathit{quasi\_standard}}) \tag{6.5}$$

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 optimise 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 next. 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 initialisation delivered weighted attributes. This procedure is not intended to be performed during runtime, because the executed distributed system should not be interrupted (abort, costs …). 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 initialisation 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 initialisation, 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

$$M_{measurement}^{assessment} \rightarrow M_{measurement}^{controlling} \tag{6.6}$$

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 (6.7) 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 (6.8).

$$f^{funct} : \text{Process model} \mapsto \{\text{Service}, ...\} \tag{6.7}$$

$$E = f^{funct}(PM) \tag{6.8}$$

Using the classic normalization approach presented in (6.9), 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.

$$v_{i,j}^{norm} = \frac{v_{i,j} - \min(v_i)}{\max(v_i) - \min(v_i)} * (\max^{norm} - \min^{norm}) + \min^{norm} \tag{6.9}$$

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 (6.10) to (6.14)). Formula (6.10) adjusts the normalized evaluation values to ensure proper calculation. If *v=1,* it describes the best quality level and no adjustments are necessary, otherwise a minimum extremum is desired and *1-v* must be calculated.

$$f^{mm}(v) = \begin{cases} v & \text{,if a maximal } v \text{ is the best} \\ 1 - v & \text{,if a minimal } v \text{ is the best} \end{cases} \tag{6.10}$$

$$f^{eval}(e_i) = \sum_{j=0}^{n-1} f^{mm}\left(v_{i,j}^{norm}\right) \Big| e_i \in E \wedge n = |QM| \tag{6.11}$$

$$V = \left\{ f^{eval}(e_i) \middle| \forall e_i \in E \right\} \tag{6.12}$$

$$e^{worst} = e_{index} \; | index = \min(\{x \mid v_x = \min(V)\}) \wedge e_{index} \in E \qquad (6.13)$$

$$E' = E \setminus e^{worst} \qquad (6.14)$$

To determine the best evaluated service, Formulas (6.11) to (6.14) are repeated until E' contains only 1 element. It provides the needed functionality and is the most appropriate one according to the specified quality model. After the service's selection it can be executed and measurement about runtime behavior will be captured to get additional quality evaluations for this service.

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 takes place. Considering the quality assurance the modified kind of measurement tools can be described as *essential measurement improvement* as

$$T^{semi\_autmatic}_{one\_meas.\_phase} \rightarrow T^{autmatic}_{one\_meas.\_phase} \rightarrow T^{automatic}_{whole\_measurement} \qquad (6.15)$$

including the derivation of the involved personnel that is used only for the first steps of infrastructure building as

$$P^{practitioner}_{measurement\_application\_staff} \rightarrow P_{initial} \qquad (6.16)$$

The last step five of the presented approaches cover the *evaluation of the entire process* as an input for the experience factory. It compares the achieved results with the desired ones.

This leads us to the unified approach of ***software measurement service infrastructure*** combining the characteristics of (6.1) and (6.2) using the explanations in (6.3) to (6.6) and (6.15) to (6.16) as

$$\begin{aligned}
&MP^{infrastructure}_{measurement\_service}: \qquad \mathbf{(6.17)}\\
&(G^{managing}_{goals\_in\_use} \times A^{product \wedge resources \wedge process}_{original}\\
&\quad \times M^{controlling}_{measurement})\,T^{semi\_automatic}_{some\_measurement}, P_{initial}\\
&\rightarrow (V^{ratio\_scale}_{data\_basis} \times U^{software\_unit}_{quasi\_standard})\,T^{automatic}_{some\_measurement}, P_{initial}\\
&\rightarrow E^{formula}_{extension} \times A^{product}_{controlled}.
\end{aligned}$$

7 Conclusions and Future Work

This technical report 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). Based on these technologies an infrastructure-based measurement service was discussed considering the quality assurance themselves. The following figure summarizes the different aspects of measurement process evaluation considering the best at the outer circle.

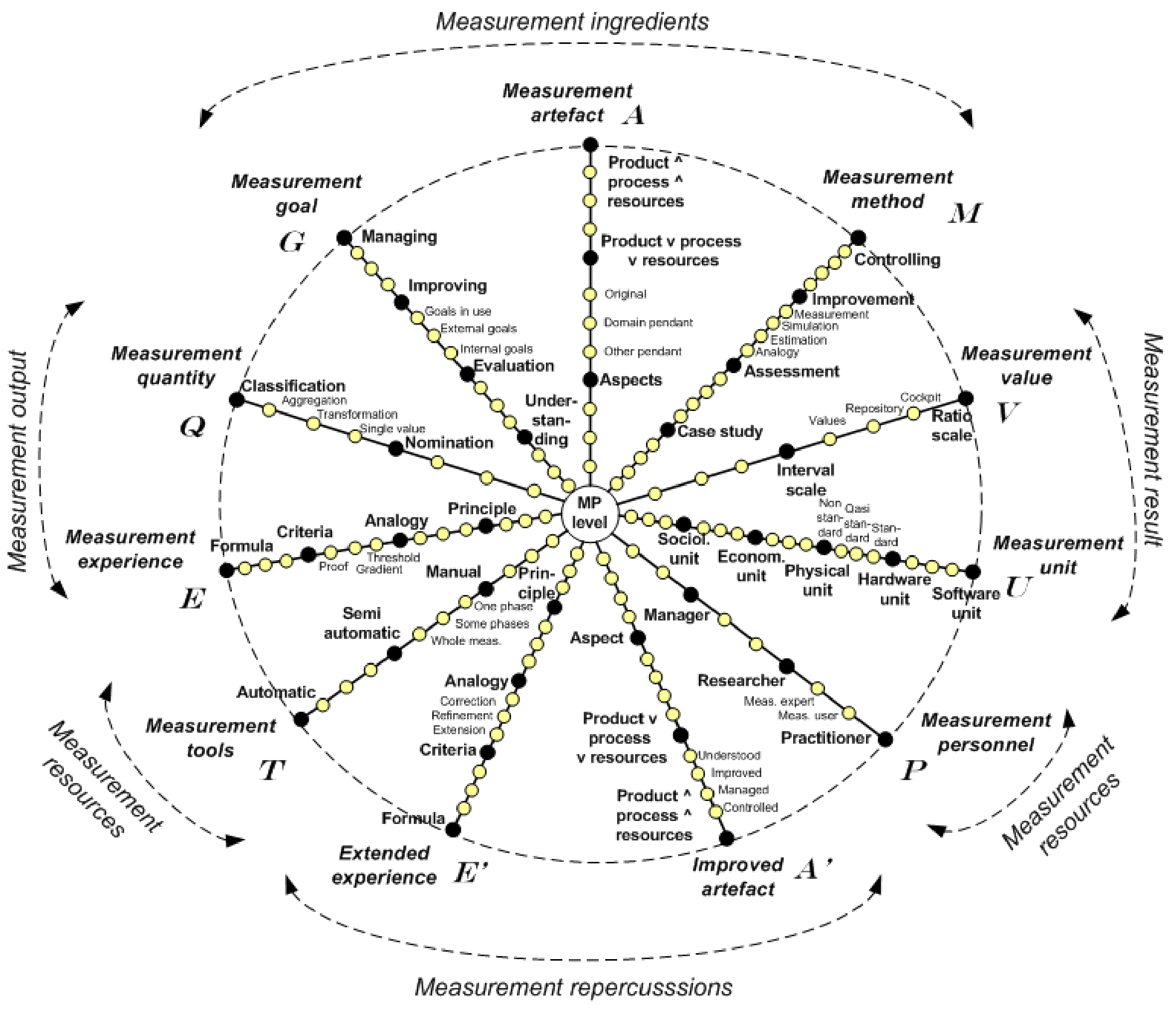


Figure 48: Software measurement process aspects and levels

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. The first figure 49 compares the measurement process levels of an example of traditional measurement, e-Measurement and AOSE.

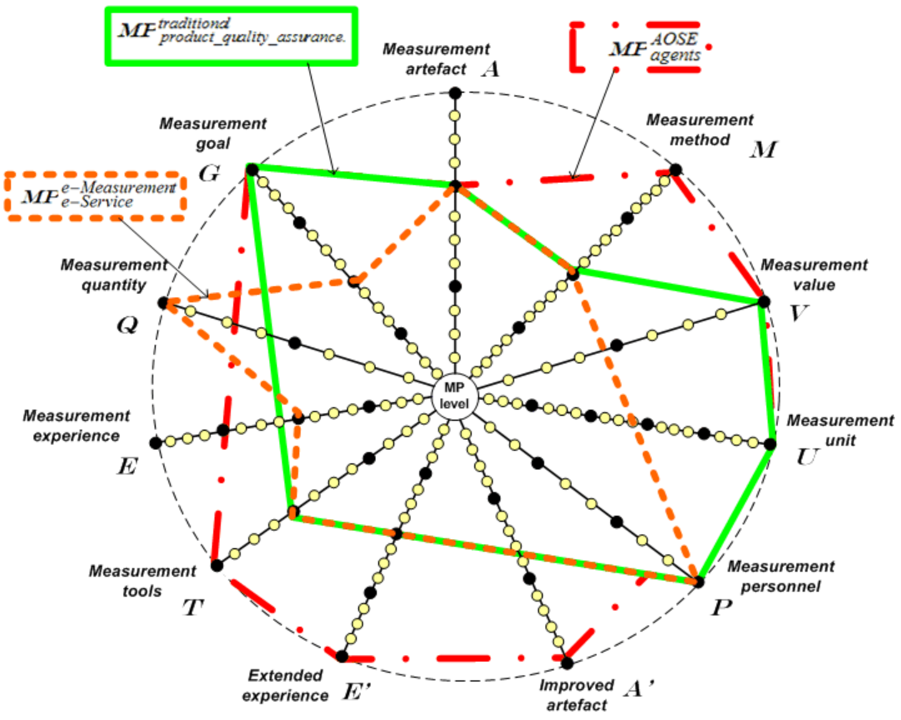


Figure 49: Examples of software measurement process levels

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. ontologies, 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.

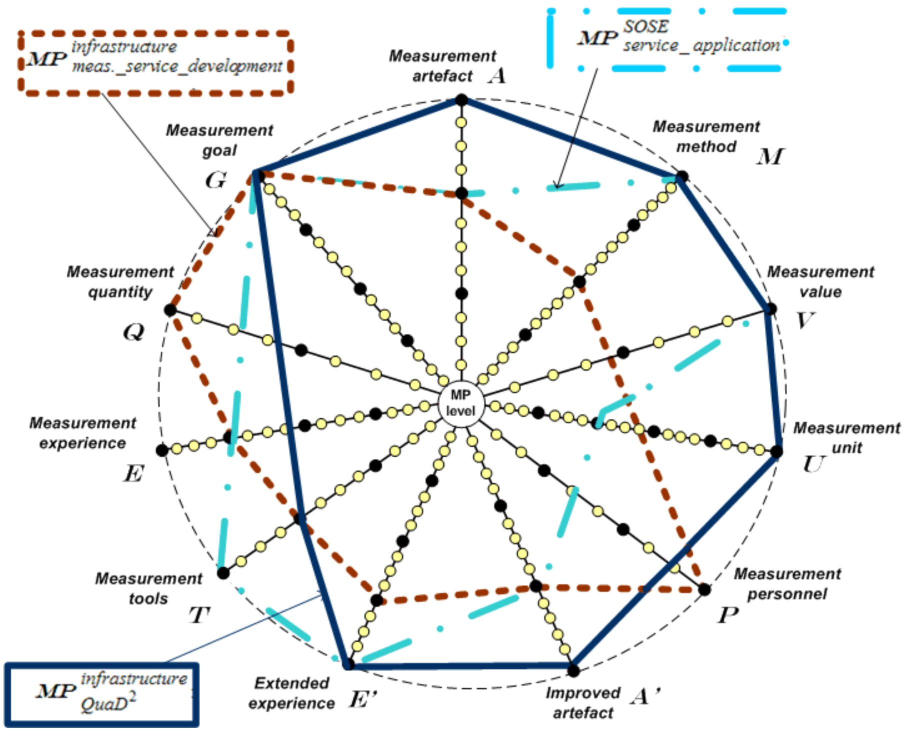


Figure 50: Software measurement process levels including the QuaD^2 approach

An implementation of this approach for specific systems is currently being performed. For the areas of software measurement infrastructures [Kunz, 2006] first components are realised. Their completion and usage may reveal opportunities for future steps.

8 References

[Abran 2006] Abran, A. et al. (Eds.): *Applied Software Measurement.* Shaker Publ., 2006

[Basili, 1994] Basili, V.R.; Caldiera, G. and Rombach, H.D.: *The Experience Factory*, Wiley & Sons, pp. 469-476, 1994.

[Basili, 1999] Basili V. R.: The Experience Factory: *Packaging Software Experiences*, In Proceedings of the NASA Goddard Space Flight Center's 14th Annual Software Engineering Workshop, ISERN-99-19 Production and Maintenance of Software Measurement Models, 1999.

[Bass 2003] Bass, L.; Clements, P.; Kazman, R.: *Software Architecture in Practice*. Addison-Wesley Professional, 2nd (hardcover) edition, 2003

[Bauer 2004] Bauer, B. and 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.

[Boehm 2007] Boehm, B. W.*: Software Engineering*. IEEE Computer Society, Los Alamitos, 2007

[Bourque 2007] Bourque, P.; Oligny, S.; Abran, A.; Fournier, B.: *Developing Project Duration Models in Software Engineering*. Journal of Computer Science and Technology, 22(3), pp. 348-357

[BPMN 2005] OMG (Object Management Group): *Business Process Modeling Notation (BPMN)*. Final Adopted Specification, 2005

[Braungarten 2007] Braungarten, R.: *The SMPI model: A stepwise process model to facilitate software measurement process improvement along the measurement paradigms*. PhD, University of Magdeburg, May 2007

[Braungarten 2005] Braungarten, R.; Kunz, M.; Farooq, A.; Dumke, R.R.: *Towards Meaningful Metrics Data Bases.* Proc. of the IWSM05, September 12-14, 2005, Montreal, Shaker Publ., pp. 1-34

[Cardoso 2006] Cardoso, J.: *Approaches to Compute Wotkflow Complexity.* In: Leymann et al.:The Role of Business Porcesses in Service Oriented Architectures. Proc. of Dagstuhl Seminar, 2006

[Ciancarini 2001] Ciancarini, P. and Wooldridge, M. J.: *Agent-Oriented Software Engineering*. In: Agent-Oriented Software Engineering, Springer, Berlin, 2001

[Dumke 1999] Dumke, R.: *A Framework for Software Measurement Evaluation*. Proc. of the IWSM'99, Lac Superieur, Quebec, Canada, September 1999, pp. 24-37

[Dumke 2005a] Dumke, R.*: Software Measurement Frameworks.* Proceedings of the 3rd World Congress for Software Quality (Vol. III), Munich, September 2005, pp. 75-84

[Dumke 2006a] Dumke, R.R.; Blazey, M.; Hegewald, H.; Reitz. D.; Richter, K.**:** *Causalities in Software Process Measurement and Improvement*. Proc. of the MENSURA 2006, November, 6-8, 2006, Cádiz, Spain, pp. 483-498

[Dumke 2005b] Dumke, R., Braungarten, R., Kunz, M., Hegewald, H. *An ISO 15939-Based Infrastructure Supporting the IT Software Measurement*. In: Büren et al.: Praxis der Software-Mesung, Shaker Pzbl., 2005, pp. 87-106

[Dumke 2006b] Dumke, R.R.; Braungarten, R.; Kunz, M.; Schmietendorf, A.; Wille, C.: *Strategies and Appropriateness of Software Measurement Frameworks*. Proc. of the MENSURA 2006, November, 6-8, 2006, Cádiz, Spain, pp. 150-170

[Dumke 2007] Dumke, R.; Braungarten, R.; Mencke, S.; Richter, K.; Yazbek, H.: *Experience-Based Software Measurement and Evaluation Considering Paradigm Evolution*. Büren et al.: Metrikon 2007 – Praxis der Software-Messung, Shaker-Publ., 2007, pp. 47-62

[Dumke 2004] Dumke, R.; Lother, M.; Schäfer, U.; Wille, C.: *Web Tomography - Towards e-Measurement and e-Control.* In: Abran et al.: Software Measurement - Research and Application, Shaker Publ., 2004, pp. 245-254

[Dumke 2003] Dumke, R.; Lother, M.; Wille, C.; Braungarten, R.; Winkler, D.: *eMeasurement – Gegenwärtiger Stand und Perspektiven.* In: Büren et al.: Software-Messung in der Praxis, Shaker Publ., 2003, pp. 135-148

[Dumke 2003a] Dumke, R.; Lother, M.; Wille, C.; Zbrog, F.: *Web Engineering.* Pearson Education Publ., 2003

[Dumke 2005c] Dumke, R.; Schmietendorf, A.; Zuse, H.: *Formal Description of Software Measurement and Evaluation*. Technical Report, University of Magdeburg, 2005 http://ivs.cs.uni-magdeburg.de/sw-eng/agruppe/forschung/ Preprints.html

[Ebert 2007] Ebert, C.; Dumke, R.: *Software Measurement – Establish, Extract, Evaluate, Execute*. Springer Publ., 2007

[Erl 2005] Erl, T. *Service-Oriented Architecture Concepts, Technology, and Design*. Prentice Hall, 2005

[Farooq 2005] 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

[Farooq 2008] Farooq, A.; Kernchen, S.; Dumke, R.R.; Wille, C.: *Web Services based Measurement for IT Quality Assurance.* In: Cuadrado-Gallege t al.: Software Product and Process Measurement. LNCS 4895, Springer-Verlag Berlin Heidelberg, 2008

[Gerber 2001] Gerber, C.: *Self-Adaptation for Performance Optimisation in an Agent-Based Information System*. In: Agent Technology for Communication Infrastructures, Hayzelden, A. L. G. and Bourne, R. A. (Editors), John Wiley & Sohns, LTD, Chichester, pp. 122-143, 2001

[Griffel 1998] Griffel F.: *Componentware – Konzepte und Techniken eines Softwareparadigmas*, dpunkt-Verlag, Heidelberg 1998

[Hanson 2003] Hanson, J.: *Coarse-grained interfaces enable service composition in SOA,* URL: http://builder.com.com/5100-6386-5064520.html

[Hiekel 2007] Hiekel, S.: *Bedeutung und Qualitätseigenschaften des Enterprise Service Bus im Kontext von serviceorientierten Architekturen.* Diploma Thesis, University of Magdeburg, Dept. of Compuer Science, 2007

[Huhns 2004] Huhns, M. N.: *Agent UML Notation for Multiagent System Design*. IEEE Internet Computing, (2004), pp. 63-71

[ISBSG 2003] Software Project Estimation *– A Workbook for Macro-Estimation of Software Developmet Effort and Duration.* Melbourne, 2003

[ISO 15939] ISO/IEC 15939: *Information Technology – Software Measurement Process*. Metrics News 6(2001) 11-46

[Jennings 1998] Jennings, N. R. and Wooldridge, M. J.: *Agent Technology - Foundation, Applications and Markets*, Springer, Berlin, 1998

[Kalepu 2003] Kalepu, S.; Krishnaswamy, S. Loke, S. W.: *Verity: A QoS Metric for Selecting Web Services and Providers.* In: Proc Fourth International Conference on Web Information Systems Engineering, 2003, pp. 131-139

[Kernchen 2006] Kernchen, S.; Farooq, A.; Dumke, R.; Wille, C.: *Evaluation of Java-Based Agent Technologies.* In: Abran et al.: Applied Software Measurement, Shaker Publ., 2006, pp. 175-188

[Kitchenham 2007] Kitchenham, B.: *Empirical Paradigm – The Role of Experiments*. In: Basili et al.: Emiprical Software Engineering, Springer-Publ., 2007, pp. 25-32

[Knapik 1998] Knapik, M. and Johnson, J.: *Developing Intelligent Agents for Distributed Systems*, McGraw-Hill, New York, 1998

[Kunz 2006a] Kunz, M.; Kernchen, S.; Dumke, R.R.; Schmietendorf, A.: *Ontology-based Web service for object-oriented metrics.* In: Abran et al: *Applied Software Measurement*, Shaker Publ., 2006, pp. 99-106

[Kunz 2008] Kunz et al:: *UnitMetrics: A Tool Support Refactoring in Agile Software Development.* IEEE IRI 2008, LasVegas, July 14-17, 2008

[Kunz 2008] Kunz, M.; Mencke, S.; Rud, D.; Dumke, R.: *Empirical-Based Design - Quality-Driven Assembly of Components.* Proceedings of the 2008 IEEE International Conference on Information Reuse and Integration (IEEE IRI-2008), July, 13-15, 2008, Las Vegas, Nevada, USA, S. 393-397

[Kunz 2006b] Kunz, M.; Schmietendorf, A.; Dumke, R.; Wille, C.: *Towards a service- oriented measurement infrastructure*. In Proceedings of the 3rd Software Measurement European Forum (Smef 2006), pp. 197-208, Rome, Italy, May 10.-12., 2006

[Laird 2006] Laird, L. M.; Brennan, M. C.*: Software Measurement and Estimation – A Practical Approach*. IEEEComputer Science, 2006

[Liu 2001] Liu, J.: *Autonomous Agents and Multi-Agent Systems - Explorations in Learning, Self-Organization and Adaptive Computation*, World Scientific Publ., Singapore, 2001

[Lother 2007] Lother, M.: *From Software Measurement to e-Measurement – A Functional Size Measurement-oriented Approach for Software Measurement.* Shaker Pul., 2007

[Lother 2004] Lother, M.; Braungarten, R.; Kunz, M.; Dumke, R.: *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, Shaker Publ., 2004, pp. 27-40

[McGovern 2003] McGovern, J.; Tyagi, S.; Stevens, M. E.: *Java Web Services Architecture*. Morgan Kaufmann, 2003.

[MacKenzie 2006] MacKenzie et al.: *Reference Model for Service Oriented Architecture 1.0.* OASIS, July 2006

[Martin 2003] Martin, M., Olsina, L. *Towards an Ontology for Software Metrics and Indicators as the Foundation for a Cataloging Web System.* 2003

[Mencke 2007] Mencke, S.; Dumke, R.: *Agent-Supported e-Learning.* Preprint No 8, Dept. of Computer Science, University of Magdeburg, 2007

[Mencke 2008] Mencke, S.; Kunz, M.; Dumke, R.: *Steps to an Empirical Analysis of the Proactive Class Schedule.* Proceedings of the 3rd International Conference on Interactive Mobile and Computer Aided Learning (IMCL 2008), April, 16-18, 2008, Amman, Jordan,

[Mencke, 2007] Mencke, S. and Dumke, R.: *A Hierarchy of Ontologies for Didactics-Enhanced E-learning*, In Proceedings of the International Conference on Interactive Computer aided Learning (ICL2007), Villach, Austria, September, 2007

[Panait 2006] 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

[Pandian 2004] Pandian, C. R.: *Software Metrics – A Guide to Planning, Analysis, and Application*. CRC Press Company, 2004

[Peltz 2003] Peltz, C. *Web Services Orchestration and Choreography*. IEEE Computer, 2003

[Perepletchikov 2008] Perepletchikov, M. Ryan, C.; Frampton, K.; Schmidt, H: *Formalising Service-Oriented Design.* Journal of Software 3(2008)2, pp. 1-14

[Richter 2005] Richter, K.: *Softwaregrößenmessung im Kontext von Software-Prozessbewertungsmodellen*. Diploma Thesis, University of Magdeburg, 2005

[Rud 2006a] Rud, D: *Qualität von Web Services.* VDM Verlag Dr. Müller, Berlin, 2006

[Rud 2007a] Rud, D.; Mencke, S.; Schmietendorf, A.; Dumke, R.: *Granularitätsmetriken für servicerientierte Architekturen.* In: Bren et al.: Praxis der Software-Messung, Shaker Publ., 2007, pp. 297-308

[Rud 2006b] Rud, D.; Schmietendorf, A.; Dumke, R.: *Performance Modeling of WS-BPEL-Based Web Service Compositions.* Proc of the SCW 2006, Sept. 18-22, 2006, Chicago, Illinois, IEEE Computer Society, pp. 140-147

[Rud 2006c] Rud, D.; Schmietendorf, A.; Dumke, R.: *Product Metrics for Service-Oriented Infrastructures.* In: Abranet al.: Applied Softwre Measurement, Shaker Publ., 2006, pp. 161-174

[Rud 2007b] Rud, D.; Schmietendorf, A.; Kunz, M.; Dumke, R.: *Analyse verfügbarer SOA-Reifegradmodelle – State of the Art.* In: Schmietendorf et al.: Bewertungsaspekte serviceorientierte Architekturen, Shaker Pul., 2007, pp. 115-126

[Schmietendorf 2007a] Schmietendorf, A.: *Eine strategische Vorgehensweise zur erfolgreichen Implementierung serviceorientierter Architekturen in großen IT-Organisationen.* Shaker Publ., 2007

[Schmietendorf 2002] Schmietendorf, A.; Dimitrov, E.; Dumke, R.: *Enterprise JavaBeans*. MITP, 2002

[Schmietendorf 2007b] Schmietendorf, A.; Kunz, M.; Dumke, R.: *Empirical analyses about the granularity of industrially used Web Services.* CONQUEST 2007, Sept, Berlin

[Skyttner 2005] Skyttner, L.: *General Systems Theory – Problems, Perspectives, Practice*. World Scientific Publ., New Jersey, 2005

[Sneed 2005] Sneed, H.: *Software-Projektkalkulation*. Hanser Publ., 2005

[Solingen 1999] Solingen, v. R.; Berghout, E.: *The Goal/Question/Metric Method*. McGraw Hill Publ. (1999)

[Tayntor 2003] Tayntor, C. B.: *Six Sigma Software Development*. CRC Press, 2003

[Thielen 2004] Thielen, M.: *Qualitätssicherun von Webservices.* Diploma Thesis, University of Koblenz-Landau, 2004

[Wang 2001] Wang, X., Chan, C. Ontology Modeling Using UML. Proc. of the 7th International Conference on Object Oriented Information Systems, 2001

[Wijata 2000] Wijata, Y. I.; Niehaus, D. and Frost, V. S.: *A Scalable Agent-Based Network Measurement Infrastructure*. IEEE Communications Vol. 38, (2000), No. 9, pp. 174-183

[Wille 2005] Wille, C.: *Software Agent Measurement Framework*. Shaker-Publ., 2005

[Wooldridge 2002] Wooldridge, M. J.: *An Introduction to Multi-agent Systems*. Wiley, 2002

[Yu 20067] Yu, Y.; Lu, J.: Fernandez-Ramil, J.; Yuan, P. : *Comparing Web Services with other Software Components.* Proc. of the ICWS 2007, pp. 388-397

[Zelkowitz 2007] Zelkowitz, M., V.: *Techniques for Empircal Validation.* In: Basili et al.: Empirical Software Engineering, Springer-Publ., 2007, pp. 4-9

[Zuse 1998] Zuse, H.: *A Framework of Software Measurement*. de Gruyter Publ., Berlin, 1998