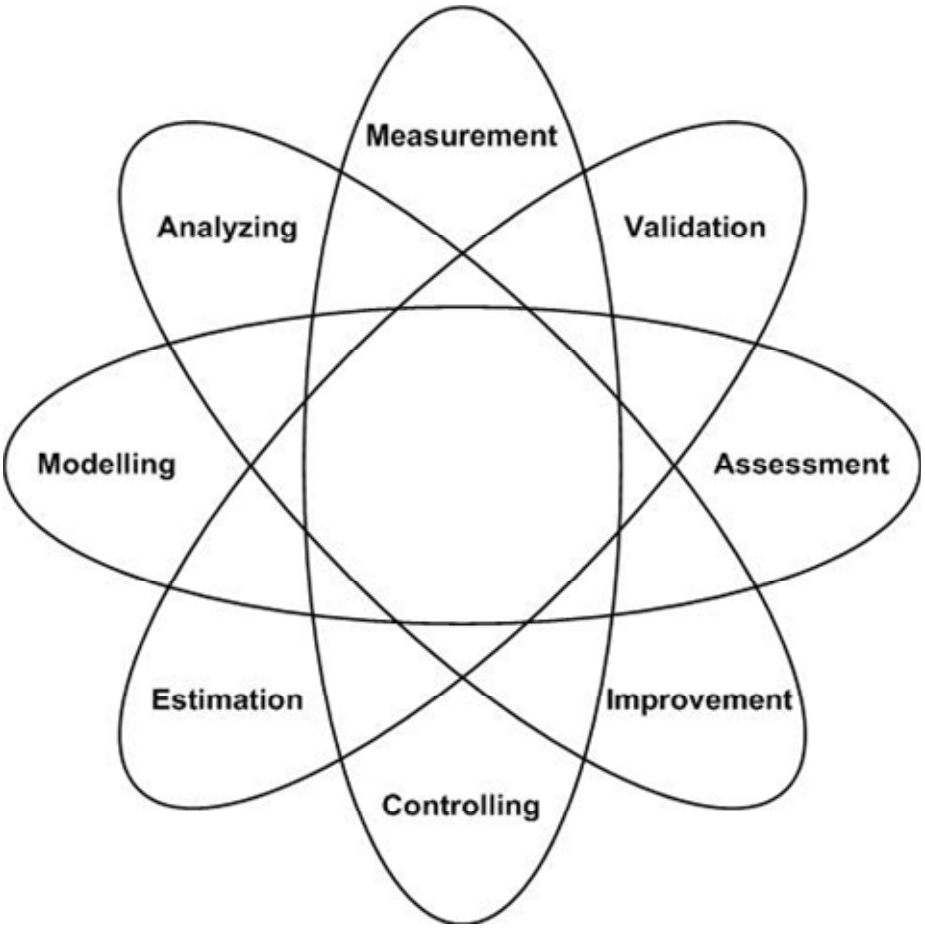
Volume 17, Number 2, October 2012

SOFTWARE MEASUREMENT NEWS

Journal of the Software Metrics Community



Editors:

Alain Abran, Günter Büren, Reiner Dumke, Christof Ebert, Cornelius Wille

The *SOFTWARE MEASUREMENT NEWS* can be ordered directly from the Editorial Office (address can be found below).

Editors:

Alain Abran
Professor and Director of the Research Lab. in Software Engineering Management
École de Technologie Supérieure - ETS
1100 Notre-Dame Quest,
Montréal, Quebec, H3C 1K3, Canada
***Tel.:** +1-514-396-8632, **Fax:** +1-514-396-8684*
aabran@ele.etsmtl.ca

Günter Büren
Vice Chair of the DASMA
Büren & Partner Software-Design GbR
Thurn-und-Taxis-Str. 12, D-90411 Nürnberg, Germany
***Tel.:** +49-911-5195511, **Fax:** +49-911-5195555*
gb@bup-nbg.de
http://www.dasma.org

Reiner Dumke
Professor on Software Engineering
University of Magdeburg, FIN/IVS
Postfach 4120, D-39016 Magdeburg, Germany
***Tel.:** +49-391-67-18664, **Fax:** +49-391-67-12810*
dumke@ivs.cs.uni-magdeburg.de

Christof Ebert
Dr.-Ing. in Computer Science
Vector Consulting GmbH
Ingersheimer Str. 24, D-70499 Stuttgart, Germany
***Tel.:** +49-711-80670-175*
christof.ebert@vector-consulting.de

Cornelius Wille
Professor on Software Engineering
University of Applied Sciences Bingen
Berlinstr. 109, D-55411 Bingen am Rhein, Germany
***Tel.:** +49-6721-409-257, **Fax:** +49-6721-409-158*
wille@fh-b1ingen.de

Editorial Office: Otto-von-Guericke-University of Magdeburg, FIN/IVS, Postfach 4120, 39016 Magdeburg, Germany

Technical Editor: Dagmar Dörge

The journal is published in one volume per year consisting of two numbers. All rights reserved (including those of translation into foreign languages). No part of this issues may be reproduced in any form, by photoprint, microfilm or any other means, nor transmitted or translated into a machine language, without written permission from the publisher.

© 2012 by Otto-von-Guericke-University of Magdeburg. Printed in Germany

Call for Participation

IWSM-MENSURA 2012

The Joint Conference of the 22nd International Workshop on Software Measurement (IWSM) and the 7th International Conference on Software Process and Product Measurement (MENSURA)

October 17-19, 2012, Assisi, Italy

THEME & SCOPE

Software measurement and metrics are key technologies for managing and controlling software development projects. Measurement is essential for any engineering activity and for increasing scientific and technical knowledge regarding both the practice of software development and empirical research in software technology. This congress will facilitate the exchange of software measurement experiences between theory and practice.

KEYNOTES:

Alain Abran:
Measurement & the Lessons from the Masters: The Benefits for Industry
Gioancarlo Succi:
Understanding the Implications of Pair Programming

THEMES of SESSIONS

- Measurement Tools and Frameworks
- Process Measurement and Improvement
- COSMIC FP Measurement and Estimation
- Efficiency in Cloud Computing
- Size and Performance Measurement
- Error Prevention Models
- Monitoring of agile Software Projects
- Effort Estimation for Web Applications
- Quality Measurement and Evaluation
- Non Functional Requirements Measurement
- Measurement Repositories

CONFERENCE ORGANIZATION

General Chair:	Luca Santillo (Agile Metrics/GUFPI-ISMA, Italy)
Organizing Chair:	Filippo De Carli (Shared Service Center/GUFPI-ISMA, Italy)
Program Chair:	Sandro Morasca (University of Insubria, Italy) Luigi Buglione (ETS/Engineering.IT, Italy)
Steering Committee:	Alain Abran (École de Technologie Supérieure, University of Quebec, Canada) Reiner R. Dumke (University of Magdeburg, Germany) and Juan J. Cuadrado-Gallego (University of Alcalá, Spain)

FURTHER INFORMATION

http://iwsm2012.wordpress.com/

Aufruf zur
Teilnahme an der

MetriKon 2012

vom 07. – 09. November 2012 in Stuttgart, Vector Consulting Services

Die Software-Metrie/Software-Metrik ist eine Teildisziplin der Softwaretechnik und hat zum Ziel, die mit Software und Softwareentwicklung verbundenen wirtschaftlich und qualitativ relevanten Aspekte quantifizierbar zu machen. Die damit verbundene Quantifizierung von Produkteigenschaften und von Beobachtungen im Ablauf von Entwicklungsprojekten und im Einsatz einer Software führt auf der Basis quantifizierter Erfahrungen zu einer deutlichen Verbesserung der Entscheidungsbasis im Projektmanagement von Entwicklungsprojekten und in den Entscheidungen zum Einsatz von Software. Wirtschaftliche Kenngrößen zur Softwareentwicklung können auf einer gesicherten Basis ermittelt werden. Grundsätzlich ist es daher für jedes Software entwickelnde Unternehmen sinnvoll, seine Arbeit und die Qualität der Arbeitsergebnisse durch geeignete Quantifizierung transparenter und besser steuerbar zu machen.

Software-Messverfahren und Metriken sind Schlüsseltechnologien für Controlling und Management des Softwareentwicklungs-Prozesses. Messen ist eine wichtige Grundlage für jede Ingenieurtätigkeit und auch für die Software-Entwicklung, sowohl in der industriellen Praxis, wie auch in der empirischen Forschung zur Software-Technik, unverzichtbar.

Diese Tagung ist mit ihrem Schwerpunkt Messen und Bewerten in der Software-Entwicklung die einzige ihrer Art im deutschsprachigen Bereich und daher seit Jahren Treffpunkt für Mess-Experten aus Industrie und Forschung. Praktiker und Entscheidungsträger können sich in einem verdichteten Programm über den aktuellen Stand der praktischen Anwendung und der wissenschaftlichen Forschung und über innovative Ansätze informieren und erhalten im intensiven Austausch Untereinander wertvolle Anregungen für die eigene betriebliche Praxis oder die Vertiefung ihrer wissenschaftlichen Arbeiten.

Wir laden Sie herzlich ein, sich an diesem bewährten Erfahrungsaustausch aktiv zu beteiligen.

Keynotes:

Christof Ebert, Vector, Stuttgart:
Technical Debt: Practical Decision-Making in Software Engineering and Its Business Relevance
Christian Lescher, TU München:
Projektmanagement-Metriken im Zeitalter der globalen Softwareentwicklung
Mario Trapp, Fraunhofer IESE, Kaiserslautern:
Messen und Bewerten funktionaler Sicherheit: Herausforderungen und Lösungsansätze

Die einzelnen **Konferenzbeiträge** befassen sich unter anderem mit folgenden Themen:

- Produktivitäts- und Wertmessung
- Qualitätsmodelle
- Funktionale Größenmessung
- Messen im Requirements Engineering
- Messen im Mobile und Cloud Computing
- Messen im Enterprise Architecture Management
- Messen im Security Engineering
- Neue Trends und Anwendungsfelder
- Messwerkzeuge

Weiterhin werden *Tutorien bzw. Workshops* von ca 1,5 Std. Dauer zur Einführung oder Vertiefung komplexer Themen und Verfahren der Software-Messung angeboten.

Außerdem ist wiederum die Teilnahme zur COSMIC-FP-Zertifizierung möglich.

Weitere Informationen zur Teilnahme finden Sie unter **www.metrikon.de**

BSOA2012

BEWERTUNGSASPEKTE SERVICEORIENTIERTER ARCHITEKTUREN

7. Workshop: 15. November 2012 in Dresden (09:30 - 17:30 Uhr)

Grußnote zum Workshop:

Prof. Dr. Reiner Dumke (Universität Magdeburg – Leiter SMLab)

Industrieorientierte Keynote:

Harry Ultsch (Geschäftsführer Zott+Co GmbH Murnau)

Praktische Aspekte der Qualitätssicherung von Cloud-Services

Wissenschaftsorientierte Keynote:

Prof. Dr. Olaf Resch (HWR Berlin)

Möglichkeiten einer Business-Rule-basierten Servicesteuerung

Moderierte Diskussionsrunde:

Dr. Frank Simon (Head of SQS Research & Innovation - SQS AG)

Mögliche Themenbereiche – Auswahl erfolgt beim Workshop:

- Service oriented Multi-Channel Architectures – the XThing
- Serviceorientierung im Bereich der analytischen Qualitätssicherung
- Serviceorientierung vom XaaS bis zum TaaS (Test as a Service)

Ausgewählte Beiträge:

Marco Mevius, Rolf Stephan, Peter Wiedmann (HTWG Konstanz & Axon Active AG Schweiz):

BPM(N)Easy – Agiles cloud- und servicebasiertes Geschäftsprozessmanagement

Klaus Lutterjohann, Daniel Simon (SQS AG):

Risikobetrachtung bei Software-Architekturen

Frederik Kramer (Uni Magdeburg):

Serviceorientierte Architekturen und Shared Services - ein konzeptueller Vergleich

Michael Gebhart (Gebhart Quality Analysis):

Effiziente Qualitätsanalyse der Gestaltung serviceorientierter Architekturen auf Basis von SoaML

Anja Fiegler, Niko Zenker, Jan Draheim, Reiner Dumke (T-Systems & Uni Magdeburg):

Referenzarchitektur von System-Management-Services in Large-Scale Cloud-Systemen

Posterpräsentationen:

Zur Anregung der Diskussion wird es innerhalb der Pausenzeiten ausgewählte Posterpräsentationen junger Absolventen und Doktoranden geben, die sich auch mit einem Kurzbeitrag im Tagungsband niederschlagen.

Felix Müller (Hochschule für Wirtschaft und Recht Berlin)

Konzepts zur Anbindung mobiler Anwendungen an Enterprise Systeme.

Leon Sczepanski (Hochschule für Wirtschaft und Recht Berlin)

ITIL/COBIT-orientierte Implementierung des Incident Managements für Cloud-Lösungen.

Ayman Massoud (Uni Magdeburg)

Efficient SOA-based Integration of Legacy Applications

André Nietze (FH Brandenburg)

Chancen zum Einsatz von Agilität in Serviceorientierten Architekturen

Marcus Zieger (Hochschule für Technik und Wirtschaft Berlin)

Einsatzszenarien ereignisgesteuerter Architekturen im E-Commerce

Ahmed A. Hussein (King Saud University, Saudi Arabia)

A Dynamic Modular Framework for Enhancing the Security of Enterprise Business Processes

Sponsoren der diesjährigen Tagung:

Fa. T-Systems Multimedia Solutions GmbH
Dresden

Fa. cimt objects ag
Berlin

Fa. Zott + Co GmbH
Murnau

Fa. T-Systems International GmbH
Magdeburg

Fa. Gebhart Quality Analysis (QA) 82
Karlsruhe

Sprecher der BSOA-Initiative:

Prof. Dr.-Ing. habil. Andreas Schmietendorf

Hochschule für Wirtschaft und Recht Berlin, FB II
Alt-Friedrichsfelde 60, D-10315 Berlin
Professur Wirtschaftsinformatik - Systementwicklung

Email: **andreas.schmietendorf@hwr-berlin.de**

Weitere Informationen und Anmeldung unter:

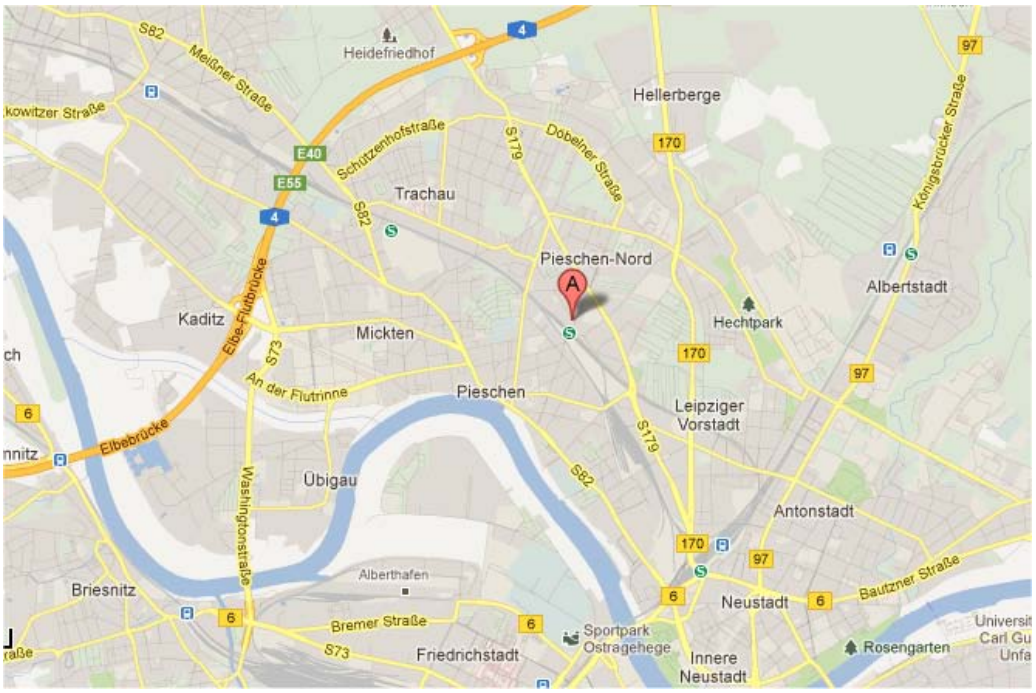
http://www-ivs.cs.uni-magdeburg.de/~gi-bsoa/2012 oder http://www.cecmg.de

Bei Anmeldung bis zum 31. Oktober 2012 wird eine Teilnahmegebühr von 90,– € (ceCMG-, DASMA-, GI- und ASQF-Mitglieder: 70,– €) erhoben, danach 110,– € (ceCMG-, DASMA-, GI- und ASQF-Mitglieder: 90,– €). Über den Tagungsbeitrag erhalten Sie eine Rechnung der ceCMG e.V. (Central Europe Computer Measurement Group).

Der BSOA-Workshop richtet sich an ein deutschsprachiges Publikum, dem entsprechend werden die Vorträge überwiegend in deutscher Sprache gehalten. Die korrespondierenden Artikel der Referenten werden den Teilnehmern in Form eines Tagungsbands zur Verfügung gestellt. Ergebnisse entsprechender Diskussionsrunden werden zeitnah im Internet publiziert.

Gastgeber des Workshops in 2012:

T-Systems Multimedia Solutions GmbH
Riesaer Straße 5, 01129 Dresden
Web: http://www.t-systems-mms.com/



Quelle: **http://maps.google.de**

Profes 2013

14th International Conference on Product Focused Software Development and Process Improvement

June 12-14, 2013, Paphos, Cyprus

CONFERENCE BACKGROUND AND TOPICS

PROFES is the premier forum for practitioners, researchers, and educators to present and discuss experiences, ideas, innovations, as well as concerns related to software process and product quality improvement. PROFES 2013 will be the 14th gathering of the PROFES community and will take place at the Azia Resort Hotel in Paphos, Cyprus **(http://www.aziaresort.com**).

The technical program will feature keynote talks, research papers, and experience reports on the most relevant topics related to process diversity management. Empirical studies (i.e., experiments, industrial experiences, cases studies, etc.) are welcome. In addition, we will host a number of workshops and tutorials. We will also host a Doctoral Symposium. Details about the program can be found on the conference website.

The proceedings of PROFES 2013 will (most likely) be published by Springer in the Lecture Notes on Computer Science (LNCS) series **(http://www.springer.com/computer/lncs**) (final approval pending).

Traditionally, PROFES has provided topics that are up-to-date, important, and interesting. To ensure that PROFES will retain its high standards and keep focusing on the most relevant research issues, we are looking for original research on software engineering techniques, methods, and technologies for product-focused software development and process improvement as well as their practical application in an industrial setting. The topics of interest include, but are not restricted to, the following:

- Product quality models for software, systems, and services
- Project management techniques
- Service orientation and Cloud
- Software development for Cyber-Physical Systems
- Evidence-based Software Engineering
- Software Process Improvement (SPI)
- Agile and lean processes
- Global software development
- Outsourcing, near-shoring, and off-shoring
- Process modeling, management, and assessment
- Processes for COTS and OSS-based development
- Service engineering processes
- Software business models
- Industrial experiences and case studies on the topics above

SUBMISSION OF TECHNICAL PAPERS AND EXPERIENCE REPORTS

All submitted papers will undergo a rigorous peer review process. Papers will be selected based on originality, quality, soundness and relevance. All contributions must be original, not published, accepted or submitted for publication elsewhere. All papers should be submitted in PDF format through Easychair (**https://www.easychair.org/conferences/?conf=profes2013**). The maximum length of the research and experience report papers is 15 pages. Submitted papers must conform to the Springer LNCS layout rules (**http://www.springer.com/computer/lncs?SGWID=0-164-7-72376-0**).

SUBMISSION OF OTHER CONTRIBUTIONS

The submission details for short papers, posters, PhD descriptions, as well as for proposals for workshops and tutorials are available at the conference website.

TENTATIVE DATES

- Paper Submission Deadline: Jan. 31, 2013
- Notification of Acceptance: Mar. 18, 2013
- Camera-Ready Version: Apr. 1, 2013

ORGANIZING COMMITTEE

General Chair:

Andreas Jedlitschka, Fraunhofer IESE, Germany

Program Committee Co-Chairs:

Markku Oivo, University of Oulu, Finland

Jens Heidrich, Fraunhofer IESE, Germany

Organizing Chairs:

George Angelos Papadopoulos, University of Cyprus, Cyprus

For a complete listing of all chairs and program committee members, please visit the conference website:

http://www.profes-conferences.org

COSMIC Annual Report 2011

“A year of great progress”

December 2011

Foreword

The COSMIC Constitution, adopted in December 2010, requires that we issue an Annual Report.

This is our first Annual Report, covering the calendar year 2011. Our aim is to inform existing and prospective users of the COSMIC Functional Size Measurement method and its related methods of the progress we have made over the past year and of our current activities and plans.

Alain Abran, Chair

Charles Symons, President

Copyright 2011. All Rights Reserved. The Common Software Measurement International Consortium (COSMIC). Permission to copy all or part of this material is granted provided that the copies are not made or distributed for commercial advantage and that the title of the publication, its version number, and its date are cited and notice is given that copying is by permission of the Common Software Measurement International Consortium (COSMIC). To copy otherwise requires specific permission.

Public domain versions of the COSMIC documentation and other technical reports, including translations into other languages can be found on the Web at **www.cosmicon.com.**

EXECUTIVE SUMMARY

The year 2011 has seen great progress in the acceptance and further development of the method. The number of downloads of our principle method definition document has more than doubled over the past year (now well over 2000[1]). There are several other indicators of great progress, in particular reports of organizations achieving major business benefits from using the COSMIC method.

Over the year, we have published two new translations of the principal method definition document which is now available in nine languages (Arabic, Chinese, Dutch, English, French, German, Italian, Japanese and Spanish) and four more translations are in progress (Polish, Portuguese, Russian and Turkish). Two new Guidelines and a new 2011 version of the ISO/IEC 19761 standard for the method have also been published. Representatives of four more countries (Ecuador, Russia, Sweden and Switzerland) were appointed to the International Advisory Council, COSMIC's main policy-making body, bringing the total to 22 countries represented on the IAC.

Important work has been done to align the COSMIC method concepts with the ISO standard that defines how to document a software architecture. This will result in some simplification of the description of the COSMIC method but no changes to the basic measurement rules.

Our guiding principles when working to enhance the method and its support offerings are, in descending order of importance:

- To ensure continuity of the measurement method so that it is 'future-proof' and all measurements will remain valid indefinitely
- To extend and improve the support for the method by widening the range of guidelines, case studies, benchmark data, etc
- To grow acceptance of the method

[1] These are direct downloads from the **www.cosmicon.com** site. Many national software metrics associations and user organizations make our documentation available on their own sites.

1 INTRODUCTION: MEASURES OF OVERALL PROGRESS

COSMIC (the Common Software Measurement International Consortium) was founded in late 1998 by a group of software metrics experts with the aim of developing a new method of software sizing based on fundamental software engineering principles, and that would be applicable to business, real-time and infrastructure software. COSMIC is an 'open', not-for-profit organization, staffed entirely by volunteer professionals from industry and academia.

The first draft definition of the method was published in 1999, field trials were conducted successfully in 2000 and the first public definition of the method was published in 2001. The end of the year 2011 therefore marks the tenth year in which the method has been publicly available.

Any new method or technology moves through a well-known series of stages of acceptance. Initially there is slow growth when it is tested by pioneers ('risk takers'), before being taken up by 'early adopters'. At some point, the new idea then either takes off with mass acceptance, or it dies. 2011 has been the year in which we can see clear signs of progressing from early adopters to mass acceptance.

- The Measurement Manual, in all its translations was reported as having had 'over 1000 downloads in late 2010. At the end of 2011, the comparable total[2] is approaching 2300, i.e. the take-up has roughly doubled in one year.
- New translations of the Measurement Manual have been published in Chinese, and German. Translations are underway in Polish, Portuguese, Russian and Turkish.
- Users of the method are not often willing to admit they are using it, and even less inclined to report their success stories. However, in 2011 in the real-time domain, Renault has published a detailed account of how it is now able to use the COSMIC method to predict the cost of its bought-in embedded software for Electronic Control Units, and how it has automated the software sizing. Saab/GM reported how it uses the method to predict the hardware memory size required for its ECU's. In the business application domain, at the end of 2011 some major financial institutions, including ING of the Netherlands are trialling the method. There have been several positive reports of using COSMIC to control projects developed using Agile methods. (A list of organizations that have publicly acknowledged their use of the COSMIC method is available for download from www.cosmicon.com.)
- The United States Government Accountability Office published a "Cost Estimating and Assessment Guide: Best Practices for Developing and Managing Capital Program Costs". The COSMIC method is listed (without any prompting from us) as one of the best practices for software sizing. In the UK, we submitted recommendations on how to control price/performance of software-intensive IT projects in the public sector to a committee of Members of Parliament. The submission was supported by the British Computer Society.
- Researchers continue to publish papers based on COSMIC measurements both in journals and at conferences.

2 NEW PUBLICATIONS

In addition to the new translations of the COSMIC Measurement Manual, the following documents were published in 2011

- An updated version of the ISO/IEC 19761 standard for the COSMIC method, to bring it in line with v3.0.1 of the method
- 'Guideline for assuring the accuracy of measurements'
- 'Guideline for the use of COSMIC FSM to manage Agile projects'

[2] These totals are imprecise and understate the true levels. In particular the totals exclude copies available from the web-sites of several national software metrics associations, including all copies of the Japanese translation which is only available from the JFPUG site. Also, many user organizations make the documentation available on their own in-house web-sites.

- ‘COSMIC function Points: theory and advanced practice’ (book of industry and research papers edited by R Dumke and A. Abran)

A ‘Guideline for measuring real-time application software’ is nearly ready for review by the Measurement Practices Committee.

3 BENCHMARKS

A new initiative was launched in collaboration with the International Software Benchmarking Standards Group to ask users of the COSMIC method to submit more project data to the ISBSG repository. Although more project data was submitted, we would still like even more data so that we can improve and extend the benchmark data to be made available to the whole community. We urge our users to submit as much project data as possible.

At year-end, the 2009 COSMIC/ISBSG Benchmark report was being updated. The new 2011 version is expected to be released during Q1 2012.

4 CERTIFICATION EXAMINATIONS

More ‘Foundation-level’ certification examinations (previously known as ‘Entry-level’) were held during the year, notably in India, Italy, Poland and Turkey. But overall there were fewer examination candidates than in 2010. It has become evident that the lack of translations of the examination paper and the whole process of managing examinations in local languages is holding back the take-up of certification exams. This subject is being explored to meet demand for Chinese language exams.

5 WEB-SITE AND COMMUNICATIONS

The **www.cosmicon.com** web-site was upgraded to make the site easier to maintain, especially for adding announcements and details of new successful certification examinees.

By year-end, 250 people had joined the ‘COSMIC Size Users’ group on Linkedin, from over 30 countries.

6 WORK OF THE MEASUREMENT PRACTICES COMMITTEE

The main task of the MPC during 2011 has been to examine how to reconcile the concepts of the COSMIC method with the ISO/IEC 42010 standard on ‘documenting a software architecture’. This arises due to the need to conform to an ISO policy of requiring consistency across its standards.

The relevance of 42010 to the COSMIC method is two-fold. First, it will enable COSMIC to simplify its definition of a software ‘layer’ and bring it more in line with common industry usage. Second, it may be desirable to re-introduce the concept of standard ‘Measurement Viewpoints’. The latter aspect is still being studied. Whatever the final outcome of this work, there will be no impact on any existing or future size measurements – the basic measurement rules are unaffected. What will improve is the characterization of a measurement so that its meaning can be clearly understood and it can be used with confidence.

When this work is completed, there will be a need to upgrade the Measurement Manual and certain Guidelines to take account of these changes and of Method Update Bulletins that have been published since release 3.0.1.

7 ORGANIZATION

During 2011, membership of the International Advisory Council was enhanced with representatives of four more countries, namely Carlos Monsalve and Monica Villavicenzio (Ecuador), Maxim Rusakov

(Russia), Kenneth Lind (Sweden) and Thomas Fehlmann (Switzerland). Jaroslaw Swierczek joined as an additional representative for Poland.

It is with great sadness that we learned of the death of Grant Rule, a member of the original Core Team that designed the COSMIC method, and a member of the MPC since its foundation.

At the end of 2011, in accordance with the Constitution, Charles Symons will step down as President of COSMIC. Luca Santillo will take over as President for a three-year term.

8 ACTIVITIES AT THE NATIONAL LEVEL

Japan held the International Workshop on Software Metrics in Nara, in November, where several COSMIC-related papers were presented. The proceedings should be published by the IEEE.

COSMIC joined with the UK Software Metrics Association to hold an 'International Conference on Software Metrics and Estimating' in London in October, where several COSMIC-related papers were presented. All papers are available from www.uksma.co.uk.

(To be completed)

9 FINANCES

COSMIC is incorporated as a not-for-profit organization in Montreal, Canada. Our only income is from certification examinations. Hitherto, we have only spent small amounts on expenses. At the end of our financial year (end October), COSMIC assets amounted to over 11,000 Euros.

10 FUTURE PLANS AND PRIORITIES

In an organization that is entirely dependent on volunteers for its progress, where the volunteers have many other commitments, it is challenging to make firm plans. Rather, we set out our intentions each year at an Annual Meeting, where IAC representatives discuss priorities for future developments. Certain principles underlie these intentions for future developments. These are, in order of importance:

a) To ensure continuity of the basic measurement method whilst improving its explanation so that it becomes easier to learn and apply consistently to ensure measurement repeatability. This principle will guide the work of the MPC in its planned next release of the Measurement Manual and of certain Guidelines, as described above.
b) To extend the support for the method. In addition to the Guideline on sizing real-time applications which is well advanced, work has started on Guidelines for project estimating, convertibility, early and approximate sizing, and mapping from UML to COSMIC. Further, research has started on the topic of accounting for Non-Functional Requirements which could lead to another Guideline in the future. Apart from Guidelines, we need to publish more case studies with worked examples of the measurement of requirements from various domains, to grow and to refine the available benchmark data, and to extend the certification examinations to an 'Advanced' level.
c) To grow the usage of the COSMIC method by improved and more extensive marketing of the method at both the senior IT management level and at the practitioner level.

There is much still to do. COSMIC would therefore welcome more volunteers to work on these various tasks. Anyone interested to contribute should contact Luca Santillo (President from 2012) via the **www.cosmicon.com** site.

Additional COSMIC News

The Quick Reference Guide (to the COSMIC method, for Business software) is a new short document for every-day reference, it is available publicly on the cosmicon portal (\guidelines\business).

At ETS under direction of Alain Abran, Sylvie's thesis on using COSMIC as a requirements improvement mechanism, and Hassan Soubra's thesis on COSMIC measurement automation in an automotive environment, have been completed and successfully presented over the past few months.

Welcome to a new national association to PSMO (Polish Software Measurement Association, **www.psmo.pl**), and congratulations and good luck to our colleagues Beata Czarnacka-Chrobot and Jarosław Świerczek. Among interesting news from Poland, the Polish Social Insurance Institution (government public institution at national level) is using the COSMIC method since over a year ago, and published an announcement on their website, specifically mentioning the choice of the COSMIC Method as the easiest/still accurate method in the field (**http://zus.pl/default.asp?p=1&id=1&idk=1940**). Also, PSMO is organizing certification exams.

The Certification Chair (Jean-Marc Desharnais), in consultation with President and Chair, is updating the Certification Rules set, including new aspects regarding providing the exam in other languages than English (interest has been shown from Poland, China, Brazil, and Turkey – exams in local languages will be allowed where the Measurement manual itself is translated, of course).

The latest COSMIC benchmarking initiative with ISBSG, an updated analysis report is available for purchase from the ISBSG website, named "The Performance of Business Application, Real-Time and Component Software Projects (2012)", at a very affordable price (now below 50 euros). Please consider purchasing and passing the news to your communities (**http://www.isbsg.org/isbsgnew.nsf/WebPages/7487DAFCD90E2CA0CA257642000FA364**).

Note that data collection (of COSMIC-measured projects, or eventually existing applications under maintenance) for benchmarking purposes is still a priority, so I warmly encourage all of you to submit and make users submitting measured projects or systems to ISBSG, no-need to wait for specific initiatives – each submission is rewarded by a contextual Benchmark Report of the submitted item against a sample of other projects in the ISBSG repository. Complete (free) questionnaires by ISBSG are available online (**http://www.isbsg.org/ISBSGnew.nsf/WebPages/286528C58F55415BCA257474001C7B48?open**)

Improvement the persistence layer inside well established software applications

Andreas Schmietendorf [1], Robert Neumann [2]

[1] Berlin School of Economics and Law, FB II – Business Information Systems,
Alt-Friedrichsfelde 60, 10315 Berlin,
andreas.schmietendorf@hwr-berlin.de

[2] Otto-von-Guericke-Universität Magdeburg, FIN-IVS,
Department of Distributed Systems, 39106 Magdeburg,
robneuma@ivs.cs.ovgu.de

Summary. *The use of Object/Relational mapping mechanism is inside greenfield projects an accepted and well established approach, in difference to brownfield projects. These types of projects are based on grown software architectures and mostly difficult to maintain. A later integration of a persistence layer should be driven carefully. Beside the selection of a corresponding framework, it is necessary to evaluate the pre- and disadvantages of integration. Over it out there is an integration procedure required under consideration of automation possibilities. The paper investigates related works and provides a generic concept for the integration of a persistence layers. The validation aspect is considered by the use of an old software solution, implemented by Microsoft technologies.*

1 Motivation and aims

Grown software systems frequently show inadequate design behaviours. Typical problems refer to following aspects:

- Insufficient use of separated architecture components,
- Mixed responsibilities inside components,
- Low abstractions of technological aspects.

Results from these architectural weaknesses are high efforts with the maintenance and development. An important task within software engineering deals therefore with the improvement of existing design approaches.

Within this paper, we want to concentrate on design improvements at an existing persistence layer. The investigation was driven by an applied research project supported by an industrial partner. Because of the industrial background, it was necessary to find a solution approach for an established software system.
There exists a so called impedance mismatch between the object oriented programming paradigm and mostly used relational database systems. Early software implementations gave only a low attention the implementation of stable data base mappings. The main research deals with the following question:
How can existing software applications profit from the use of modern persistence layer mechanism?

Especially we want to address the following topics:

- Evaluation model for persistence layer mechanism,
- Possibilities for source code generation,
- Measurements based evaluation approaches,
- Quality assurance for changed source code.

With the help of the earned knowledge want to derive a generic design guideline to change existing persistence layer approaches.

2 Related works

A very good description of possible approaches for the implementation of database connections can be found by [Ambler 2005]. He distinguishes 3 basic approaches in this context:

- Embedded SQL in domain/business classes (easy to implement, but difficult to maintain),
- Creating data classes corresponding to domain/business classes (better encapsulation of database access, but problems with database changes)
- Robust persistence layer that maps objects to persistence mechanisms (object-oriented implementation without database knowledge)

The robust persistence layer approach deals with an optimized decoupling of the scheme inside the database system and the object oriented implementation. On this basis the software developer should be independent from storage details inside the database system.
A well known approach can be found by the use of data transfer objects (short DTO). This approach was first coined by [Fowler 2003]. DTOs are responsible for the provision of data objects from the database in a generic way. There is no business functionality assigned. The implementation of DTOs can be driven in different ways. As shown by [Allen 2010] there are tree important pattern for the persistence layer, cited under application from [Fowler 2003]:

- Repository Pattern – “mediates between the domain and data mapping layers using a collection-like interface for accessing domain objects”
- Unit of Work Pattern – “maintain a list of objects affected by a business transaction and coordinates the writing out of changes and the resolution of concurrency problems”
- Lazy Load Pattern – “an object that doesn’t contain all of the data you need but knows how to get it”

A technology driven discussion of the persistence layer application can be found under [Matytschak 2005]. The paper shows specific requirements aspects for the selection of usable persistence solutions.

3 Concept of an improvement approach

3.1 Selection criteria for Object Relational Mapper

The checklist provided by [Matytschak 2005] proposes to consider 17 evaluation aspects. Many aspects depend from the internal implementation of the persistence layer framework. From our point of view, there are two major sources for selection criteria. First the provided functionalities of a given mapping frameworks and second the specific conditions from considered applications. Some examples for evaluation aspects:

Functional evaluation aspects:

- Supported database systems,
- Performance behaviour and caching aspects,
- Lazy Loading aspects,
- Possible degree of automation.

Application related aspects:

- Available migration strategies,
- IDE Integration/visual tools,
- Use of existing database implementations,
- Required transactional behaviour.

Economical aspects:

- Licence and maintenance costs,
- Maturity of the framework,
- Available Experiences,
- Provider standing in the market.

Beside the shown requirements there are to consider the providers capability or open source possibilities. Furthermore there are knowledge prerequisites of the own employees.

3.2 Expected improvements (Measures)

Derived from real problems inside existing software application there are concrete expectations by the use of a persistence layer. Improvements can be expected with the internal architecture especially the independence of the software implementation from the underlying database system.

- Decoupling the business logic from the persistence mechanism,
- Give up the use of Store Procedures,
- Give up the use of embedded SQL,
- Visual supported generation of access methods,
- Strong typing of SQL statements (e.g., Linq),
- Reduction of the mistake susceptibility.

The use of a persistence layer implies some possible disadvantages too. Very often are mentioned problems with the reachable performance, the usability inside brownfield approaches or the handling of generated source code fragments. Another important challenge refers to necessary test activities.
The best possibility to the check up reached improvements would be the application of corresponding metrics. During the preparation of this contribution we analyzed the possibilities of the following measurement approaches:

- LoC within the Stored Procedures vs. LoC within the data access layer (short DAL): How did the size of the stored procedures reduce and how is this reflected in the DAL?
- LoC generated code within the DAL (by the use of mapping frameworks) vs. manual written LoC inside the DAL

Further investigations try to use cohesion and coupling metrics as well as transferable aspects from the ripple effect. As proposed though [Bilal 2006]: „The ripple effect metric shows what impact changes to software will have on the rest of the system."

3.3 Procedure for changes

Figure 1 shows a raw description of the change process. A real change should be prototypical driven. That means, after the implementation of a small part experiences should be derived for the next steps and finally used to define the whole change procedure. This process corresponds with selected aspects of the agile software development.

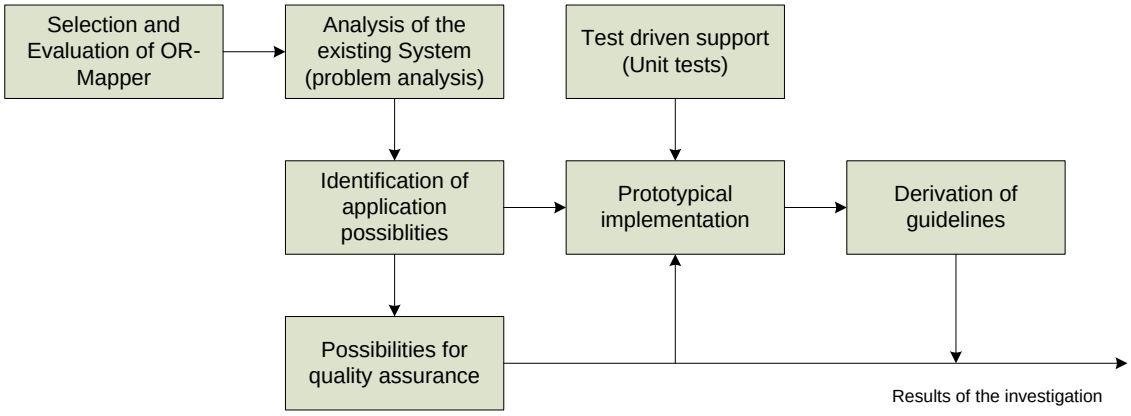


Figure 1: Preparation of the change process

4 Prototypical aspects

4.1 Available persistence frameworks

There exist various vendors of persistence frameworks for very different platforms. The following represents a non-exclusive list of persistence frameworks available in the Microsoft related market:

- AgileFx (http://www.agilefx.org/),
- nHydrate (http://www.nhydrate.org/),
- Signum Framework (http://www.signumframework.com/),
- SubSonic (http://subsonicproject.com),
- ADO.NET Entity Framework (http://msdn.microsoft.com),
- NHibernate (http://nhforge.org).

A first analysis has shown difficulties with the first four approaches. The problems referred to an insufficient maturity, inadequate support or restricted support of database systems. For the prototypical tests, a concentration took place on the two last approaches.

4.2 Examples of applications

The industry project that we were working on is based on the Microsoft .Net stack and is implemented in C#. An own custom data access layer already existed, but used embedded SQL for data access. In the database, store procedures filled query structures with the query results, which were returned to the C# DAL client.

Since in the .Net world, nHibernate and the ADO.Net Entity Framework (EF) represent the only persistence frameworks that –due to their good standing in the market– qualify for application in enterprise projects, the following analysis bases on the two before mentioned alternatives.

Both nHibernate as well as EF provide support for Microsoft's Language-integrated Query (LINQ), a C# language extension that makes it possible to author SQL-like statements in native C#. While nHibernate has been on the market for a very long time, EF was released by Microsoft only a few years ago. In its first versions, EF especially suffered from performance-related problems, which significantly hindered its adoption through customers. Nowadays, EF has reached a very high maturity and can be considered as real alternative to Hibernate. Both nHibernate and EF provide in-built support for Lazy Loading, which is especially useful, when large database tables queried. nHibernate, other than EF, furthermore has a caching feature that allows for storing frequently queried entities client-side. If entity caching was to be used from within EF, an external caching system would need to be used.

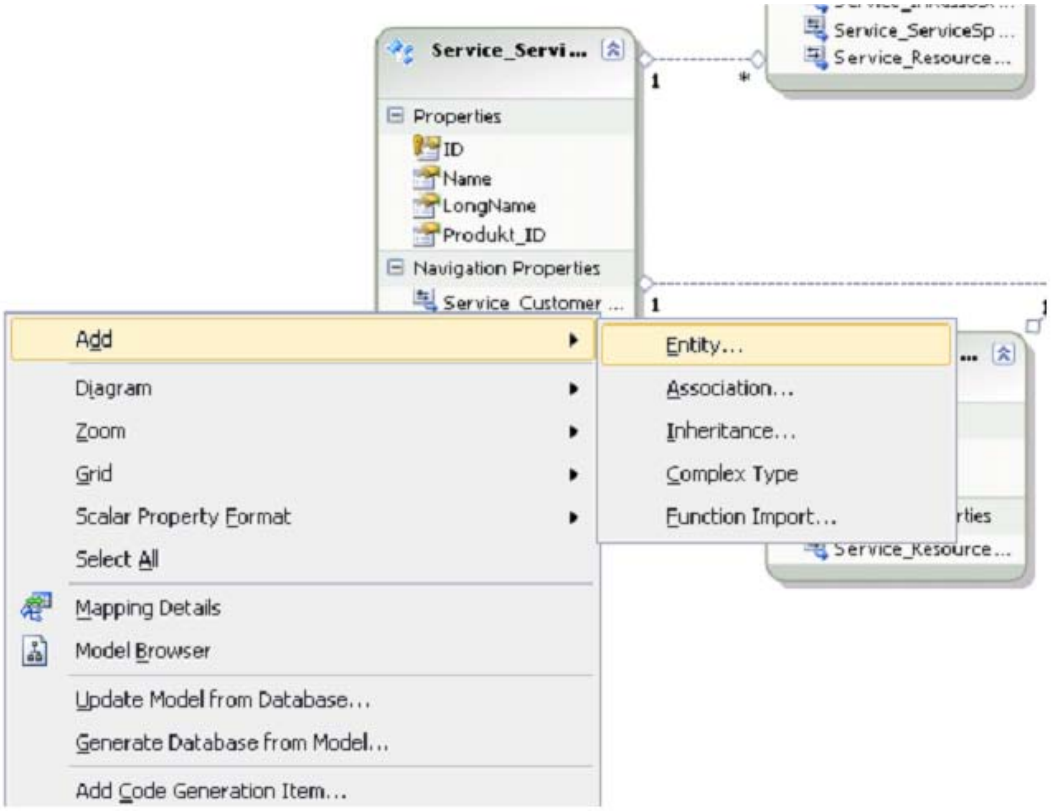


Figure 2: ADO.NET Entity Data Model and Entity Designer

As a Microsoft product, EF and all related tools seamlessly integrate with the Visual Studio IDE. A powerful designer (see figure 2) allows for designing entities from within Visual Studio. A Wizard generates entity data contexts and automatically creates code proxies inside the solution project.

(Free) wizard and designer tools (e.g., nHibernate Mapping Generator) also exist for nHibernate, but do not integrate with the IDE. We found the Mapping Generator difficult to use and error prone. The 80 generated mapping files were partially wrong, and had to be manually adjusted. Creating the mapping information consumed significantly more time with nHibernate than with EF.

Another interesting feature of EF is the native support for WCF data services. Data services virtually expose data contexts through a WCF service and imitate a real data context client-side. This powerful feature was missing in nHibernate.

4.3 First findings

Many papers describe technological possibilities of provides entity frameworks. It's currently very hard to transfer these experiences for established software systems. These applications come with difficulties in the architectural behaviours as well as used service products. The usability of a modern persistence layer requires layered software architecture. Furthermore there are strong requirements for Unit-Tests to validate a realized change against the developer specification. Surprising is the low attention of software metrics when changing data base bindings.

The current condition of our examinations can be characterized as follows (product related results – ADO.Net Entity Framework (EF)):

- Separation of the internal database schema from the used data structure inside the application,
- Fast realization of database connections, possibilities for automated configuration and database mappings,

- Use of visual tools for implementations and changes of the conceptional database model, entities and relationships,

5 Conclusions and Outlook

The submitted paper describes on ongoing work, the project will be finished only in September 2012. Therefore the chapters don't still offer any final representation of the project findings. Especially in the field of software metrics we expect a strong improvement. On part will be address applicable metrics to describe the structural behaviour of persistence layers an other measurement aspect deals with quality assurance during the change process.

References

[Allen 2010] Allen, S.: Testability and Entity Framework 4.0, Whitepaper Microsoft, 2010 (http://msdn.microsoft.com/en-us/library/ff714955(d=printer).aspx)

[Ambler 2005] Ambler, S. W.: The Design of a Robust Persistence Layer For Relational Databases, Ambysoft Inc., 2005 (**http://www.ambysoft.com**)

[Bilal 2006] Bilal, H.; Black, S.: Computing Ripple Effect for Object Oriented Software, in Proc. QAOOSE workshop (2006), Universita della Svizzera italiana, 2006

[Fowler 2003] Fowler, Martin. Patterns of Enterprise Application Architecture. Addison-Wesley, 2003

[Köker 2008] Köker, P.: Persistenz Layer Tutorial, JDBC Persistence Layer Home Page, (Abruf: **http://jdataset.de/**, April 2012)

[Matytschak 2005] Matytschak, M.: Entscheidungsgrundlagen für den Einsatz von Tools Was müssen Persistenz-Lösungen können?, dotnetpro 5/2005

[Trivadis 2012] Grundlegende Architekturkonzepte zur Erstellung von Individualsoftware, Trivadis Whitepaper, Basel (download: **http://www.trivadis.com/**)

Software-Messung und -Bewertung - Eine Bilanz

Reiner R. Dumke
(www.smlab.de)

Die Problematik der Ausbreitung von Software in wirklich allen Teilen des Lebens mit den damit verbundenen neuen Möglichkeiten aber vor allem auch zunehmenden Gefahren macht das Qualitätsproblem von Software bzw. die genaue Kenntnis seiner Eigenschaften und Prozesse zu einer zentralen Aufgabe der Software-Entwicklung vielleicht sogar der Informatik insgesamt. In diesem Beitrag sollen ausgehend von ausgewählten historischen, inhaltlichen und zweckbezogenen Beschreibungen der Stand zur Software-Messung und -Bewertung kurz skizziert, einige aktuelle Probleme identifiziert und vor allem zu weiteren Arbeiten motiviert werden.

Ein kurze Historie

Die Software-Messung - zunächst nur unter dem Schlagwort *Metriken* entstanden - wird zumeist mit dem Büchern aus den 70-er Jahren von *Frederick Brooks* (The Mythical Man Month), *Ben Shneiderman* (Software Psychology) und *Maurice Halstead* (Software Science) sowie dem Artikel von *Tom McCabe* (A Complexity Measure) in ihrer Entstehung identifiziert ([Brooks 1975], [Halstead 1977], [McCabe 1976], [Shneiderman 1980]). Dabei ging es vor allem um eine ***empirische Bewertung von Programmen*** bzw. von Software-Produkten überhaupt. Vor dem Hintergrund einer auf der Theorie formaler Sprachen basierenden Programmübersetzung und -interpretation ergab sich dabei beispielsweise eine veränderte, komplexere Betrachtung wie die folgende:

"Gegeben sei ein Programm P in der Programmiersprache L
zu einem Algorithmus bzw. einer Spezifikation S" oder kurz $S \rightarrow P_L$

wurde zu

"Ein vom Entwickler W mit der Ausbildung A und der Erfahrung E von der Abteilung I der Firma F entwickeltes Programm P in der Programmiersprache L mit der Programmbibliothek B in der Programmierungszeit T und der Entwicklungsmethodik M zum gegenwärtigen (modernen) Paradigma G, dem Testverfahren V, mit der Hardware H vom Hersteller R aus einem zugrunde gelegten Algorithmus S mit der Komplexität K und einer Qualität Q für den Kunden U" oder nicht ganz so kurz

$$S_K \xrightarrow{W(F,I,A,E),M(G,V),H(R),T} P_{L(B),Q(U)}.$$

Um Aussagen zum Umfang, zur Art des Programms bzw. seiner Komplexität zu objektivieren wurde nach "auszählbaren" Eigenschaften gesucht, die die Vielfältigkeit der Programmelemente, die strukturellen Eigenschaften oder einfach nur die Programmzeilenanzahl betraf [Zuse 1999].

Im Kontext einer großen Vielzahl von entwickelten Programmiersprachen, -techniken sowie der Programmierart selbst ergab sich natürlich auch eine Vielzahl von Metriken, die bereits die Tausend weit überschritt. Neben diesen Metriken als "Kennzahlen" ergab sich bezüglich ihrer (statistisch belegbaren) Bewertungsformen die Frage nach einem wirklichen ***Software-Maß*** (mit der zumeist verhältnisskalierten Eigenschaft und einer ***Maßeinheit***). Hierzu sind insbesondere die Bücher der 90-er Jahre von *Norman Fenton* (Software Metrics), *John Munson* (Software Engineering Measurement), *Martin Shepperd* (Foundation of Software Measurement), *Scott Whitmire* (Object-Oriented Design Measurement) und *Horst Zuse* (Software Complexity) zu nennen ([Fenton 1997], [Munson 2003], [Shepperd 1995], [Whitmire 1997], [Zuse 1991,1998]). Das schließt auch die Validation dieser Maße ein.

Als Software-Messung und -Bewertung entwickelte sich die "Metriken-Anwendung" immer mehr zu einem komplexen Prozess. Damit wurde die Art einer Metriken-Bildung mit ihrer Skaleneigenschaft und der damit möglichen Nutzung für Bewertungen und schließlich für (Software Engineering) Entscheidungen zu einem umfassenden Gebiet und erforderte somit auch eine umfassende Darstellung in ***Software-Messungs-Büchern***, wie beispielsweise die von *Barbara Kitchenham* (Software Metrics), *Capers Jones* (Applied Software Measurement), *Tom Gilb* (Software Metrics), *Paul Goodman* (The Practical Implementation of Software Metrics), *Lem Ejiogu* (Software Engineering with Formal Metrics), *Reiner Dumke* (Softwareentwicklung nach Maß) und *Stephen Kan* (Metrics and

Models in Software Quality Engineering) ([Dumke 1992], [Ejiogu 1991], [Gilb 1977], [Goodman 1993], [Jones 1991], [Kan 1995], [Kitchenham 1996]).

Die zunächst vornehmliche Orientierung auf das Programm bzw. die Software allgemein wurde zunehmend durch eine ganzheitliche Sichtweise der Software-Entwicklung mit den involvierten Hilfsmitteln und dem Personal selbst ergänzt. Es entstand eine Analyse, Messung und Bewertung der Software-Produkte, der Prozesse und den damit verbundenen Ressourcen. Speziell für die ***Software-Prozessmessung*** entstanden ganzheitliche Darstellungen von *William Florac* (Measuring the Software Process), *Lawrence Putnam* und *Ware Myers* (Five Core Metrics), *Ravindranath Pandian* (Software Metrics), sowie für ***Reviews und Audits*** von *Charles Hollocker* (Software Reviews and Audits Handbook) und für die ***Aufwandschätzung*** von *Barry Boehm* (Software Cost Estimation with COCOMO II), *Charles Symons* (Software Sizing and Estimating) und *Manfred Bundschuh* und *Carol Dekkers* (The IT Measurement Compendium*)* ([Boehm 2000], [Bundschuh 2008], [Florac 1999], [Hollocker 1990], [Pandian 2004], [Putnam 2003]). Eine Art der ***Software-Ressourcenmessung*** stellt das Buch von *Watts Humphrey* (Introduction to the Personal Software Process) zur Bewertung des Entwicklungs-personals dar [Humphrey 2000].

Die besondere Anwendung statistischer Verfahren wurde in den Büchern von *Nocer Singpurwalla* und *Simon Wilson* (Statistical Methods in Software Engineering*), Natalia Juristo* und *Ana Moreno* (Basics of Software Engineering Experimentation) sowie *Claes Wohlin* und andere (Experimentation in Software Engineering) und speziell als ***Statistische Prozess-Kontrolle*** (englisch SPC) von *Florac* und *Pandian* beispielhaft beschrieben ([Florac 1999], [Juristo 2003], [Pandian 2004], [Singpurwalla 1999], [Wohlin 2000]).

Mess- und Bewertungsverfahren werden in ihrer Qualität und Anwendbarkeit vor allem durch die zugrunde liegenden ***empirischen Erfahrungen*** bestimmt. Einige Bücher widmen sich diesem Aspekt in relativ allgemeiner Form, wie *Alan Davis* (201 Principles of Software Development), *Albert Endres* und *Dieter Rombach* (A Handbook of Software and Systems Engineering), *Jessica Keyes* (Software Engineering Handbook) sowie *Tom DeMarco* und *Timothy Lister* (Wien wartet auf Dich (englisch: Peopleware)) oder sehr konkret: beispielsweise als Sammlung von Projekterfahrungen und dessen Merkmale durch die *International Software Benchmark Standard Group (ISBSG)* (ISBSG Software Project Estimation) bzw. als systematische Übersicht in *Harry Sneed* und andere (Software in Zahlen) ([Davis 1995], [DeMarco 1991], [Endres 2003], [ISBSG 2003], [Keyes 2003], [Sneed 2010]).

Außerdem erfolgte eine immer detailliertere Analyse und Herleitung von Metriken für die verschiedensten Programmiersprachen und zum Teil auch -techniken. Wurden zunächst für prozedurale Sprachen umfangreichen Metriken-Ansätze entwickelt (siehe [Zuse 1991]) so wurden immer mehr ***objektorientierte Programme (OOP)*** bzw. die Objektorientierung generell (als ***OOSE***) untersucht. Neben den allgemeinen Prozessbezogenen Darstellungen von *Brian Henderson-Sellers* (Object-Oriented Metrics), *Scott Whitmire* (Object-Oriented Design Measurement) und *Mark Lorenz* (Object-Oriented Metrics) sind auch detaillierte Betrachtungen zu OO-CASE-Tools von *Michele Lanza* und *Radu Marinescu* (Object-Oriented Metrics in Practice) oder zu (Java-) Programmbibliotheken von *Ayaz Farooq* (Conception and Prototypical Implementation of a Web Service as an Empirical-based Consulting about Java Technologies) vorgenommen worden ([Farooq 2005], [Henderson-Sellers 1996], [Lanza 2006], [Lorenz 1994], [Whitmire 1997]).

Neben den gängigen statistischen Verfahren bzw. skalenbezogenen Abbildungsbetrachtungen wurden zunehmend ***Erfahrungen bzw. Gesetzmäßigkeiten anderer Fachgebiete*** z. B. aus der Physik oder Soziologie in die Software-Messung- und -Bewertung einbezogen. Typische Beispiele für derartige Betrachtungen sind für die Analyse und Bewertung von Web-Systemen die Bücher von *Philip Tetlow* (The Web's Awake) und *Bernardo Hubermann* (The Laws in the Web) bzw. speziell für den e-Commerce und Web Service Bereich von *Dieter Masak* (Digitale Ökosysteme) ([Huberman 2001], [Masak 2009], (Tetlow 2007]).

Der zunehmende Kenntnisstand zur Software-Messung und -Bewertung führte zu eigenständigen Prozessabläufen bzw*.* zu ***Vorgehensmodellen für die Software-Messung*** selbst. Nachdem zunächst viele Metriken in CASE-Tools ergänzt wurden, ergab sich durch die intendierte Anwendung dieser Tools auch eine (Un-) Menge von Messdaten, deren Auswertungsaufwand den wirklichen Nutzen hilfreicher Entscheidungsfindungen bei weitem überstieg. Daher wurde folgerichtig von Dieter Rombach und Victor Basili eine Vorgehensweise als ***Goal-Question-Metric*** (GQM) - auch als GQM-Paradigma bezeichnet - postuliert, die den Messansatz wieder "auf die Füße stellte" und die eigentlichen (Mess- und Auswertungs-) Ziele an den Anfang stellten ([Basili 1994], [Solingen 1999]).

Um den Vorwurf einer lokalen Optimierung zu umgehen, wurde GQM schließlich als GQM$^{+Strategy}$ in einen Business-Kontext und somit einer ganzheitlichen Zielführung, eingeordnet [Basili 2007a]. Nachdem bisher die Software-Messung und -Bewertung vor allem innerhalb der Qualitätssicherung in den Prozessmodellen und -Standards wie ISO 9001, CMMI, SPICE usw. mehr oder weniger implizit definiert war, wurde nun auch ein expliziter ***Messstandard*** (die ISO 15939) entwickelt, der erstmals den gesamten Ablauf und den Kontext einer IT-Bereichsorientierten und empirisch basierten Software-Messung und -Bewertung zum Inhalt hat ([Emam 1998], [ISO 15939 2001], [Kenett 1999], [Kulpa 2003]). Eine ausführliche, beispielhafte Erläuterung dieses Standards wurde durch *John McGarry* und andere in einem Buch zusammen gefasst [McGarry 2002].

Natürlich wurde der ***Ausbildung zur Software-Messung*** selbst entsprechend Rechnung getragen beispielsweise in den Büchern von *Linda Laird* und *Carol Brennan* (Software Measurement and Estimation) und *Horst Zuse* (A Framework of Software Measurement) ([Laird 2006], [Zuse 1998]). Insbesondere das letztere Buch mit einem e-Learning-Tool zu über 1500 Metriken, ihren Skaleneigenschaften und IT-Anwendungsbeschreibungen unterstützt in sehr detaillierter Form eine messtheoriegerechte Software-Engineering-Ausbildung.

Zum Abschluss dieser sehr verkürzten Übersicht zu einigen Etappen der Software-Messung und -Bewertung soll darauf verwiesen werden, dass für die Verbreitung und spezifische Anwendung dieses Kerngebietes des Software Engineering die ***Publikationen in den jeweiligen Landessprachen*** eine wichtige Rolle spielen. Als Beispiel seien hier nur genannt die italienische Darstellung durch *Luigi Buglione* (Misurare il software), die französische Buchausgabe durch *Pierre Robillard* (Le Logiciel: de sa conception à sa maintenance), die deutschen Publikationen durch *Georg Thaller* (Software-Metriken – einsetzen, bewerten, messen) oder auch *Christof Ebert* (Software-Metriken in der Praxis) und schließlich eine russische Beschreibung durch *Anton Isozimov* (Metricheskaja Ozenka Katschestva Programm) ([Buglione 2003], [Ebert 1996], [Isozimov 1989], [Robillard 1985], [Thaller 1994]).

Von Software-Metriken und -Maßen zu Software-Messprozessen

Zur Anwendung von Metriken belegen bzw. motivieren etliche Zitate, dass man um das Messen einfach nicht herumkommt bzw. ohne Software-Messung gar kein Verständnis von Software und Software-Entwicklung möglich sei. Es seien nur genannt: "To measure is to know" (Maxwell); "A science is a mature as its measurement tools" (Pasteur); "You cannot control what you cannot measure" (DeMarco); "If software development is to viewed as an engineering discipline, it requires a measurement component that allows us to better understand, evaluate, predict and control the software process an product" (Basili).
Natürlich gibt es auch mehr oder weniger ernst gemeinte "Gegenansichten", wie beispielsweise: "Wer viel misst misst viel Mist" (anonym); "Not everything that counts can be counted, not everything that is counted counts." (Einstein) oder gar "Our inability to actually measure knowledge means that much of our metric process is built on a foundation of sand" (Armour).

Problematischer allerdings sind Fragen zur Anwendung von Metriken, wenn sie deren Eigenschaften oder Anwendung restriktiv zuschneiden bzw. eingrenzen, wie beispielsweise:

- Software-Metriken sollten einfach anwendbar, leicht verständlich und sehr aussagekräftig sein!
- Wieviel bzw. welche Metriken sollte ich anwenden?

Hier kann man nur den Standpunkt von Humphrey unterstreichen, der die Nichtanwendung von Metriken oder unzulässige Einschränkung als typisches "out of control" für die jeweiligen Artefakte und Prozesse charakterisierte.

Wenn man allerdings bedenkt, was man alles messen könnte/sollte, so wird einem die schier unlösbare Aufgabe einer einheitlichen, für alle (sinnvoll) anwendbare Metriken- bzw. Maß-Anzahl deutlich. Allein beim Software-Produkt hätten wir beispielsweise die Artefakte bzw. Ausprägungen wie sie in Abbildung 1 angedeutet sind ([Sommerville 2010], [Sneed 2005]).

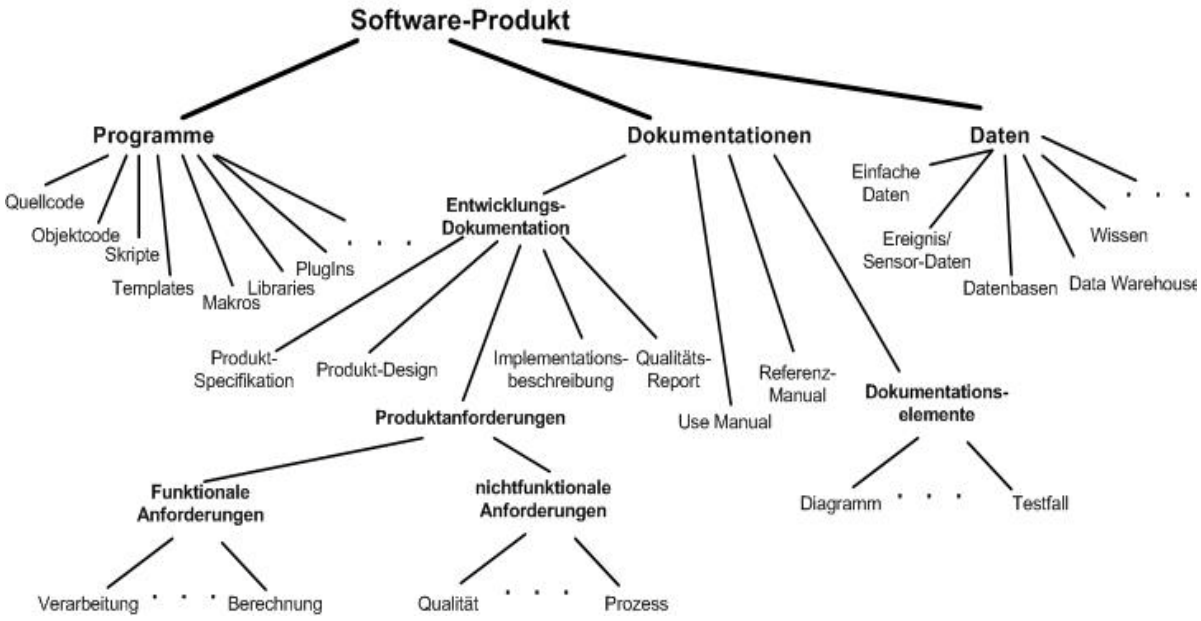


Abbildung 1: Artefakte eines Software-Produktes

Für die Software-Entwicklung bzw. die involvierten CASE- und Anwendungsgebieten ergibt sich schließlich die Ausprägung der Artefakte unseres obigen Programmentwicklungsbeispiels wie folgt ([Boehm 2007], [Sommerville 2010]).

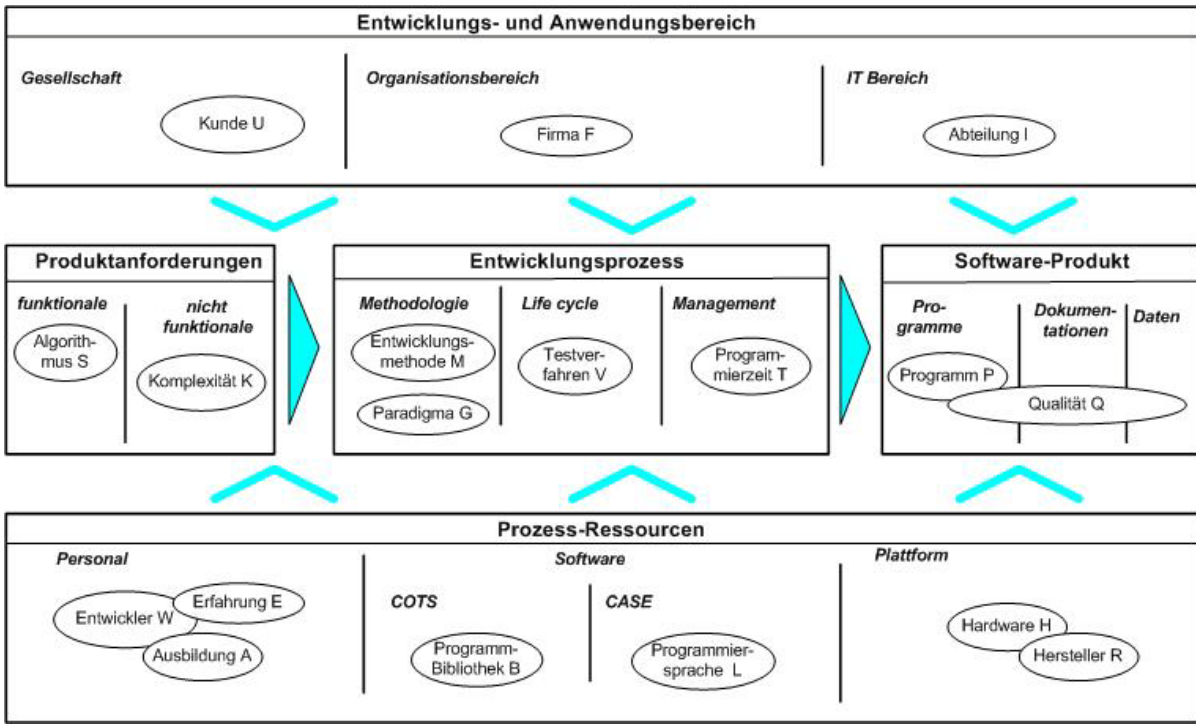


Abbildung 2: Mögliche Artefakte im Kontext der Software-Entwicklung

Bedenkt man schließlich, dass mehr oder weniger kausale Zusammenhänge zwischen diesen Artefakten bestehen, wird die Komplexität eines *quantitativen geleiteten Entwicklungsprozesses* besonders deutlich (siehe Abbildung 3 und [Dumke 2010], [Pearl 2000]).

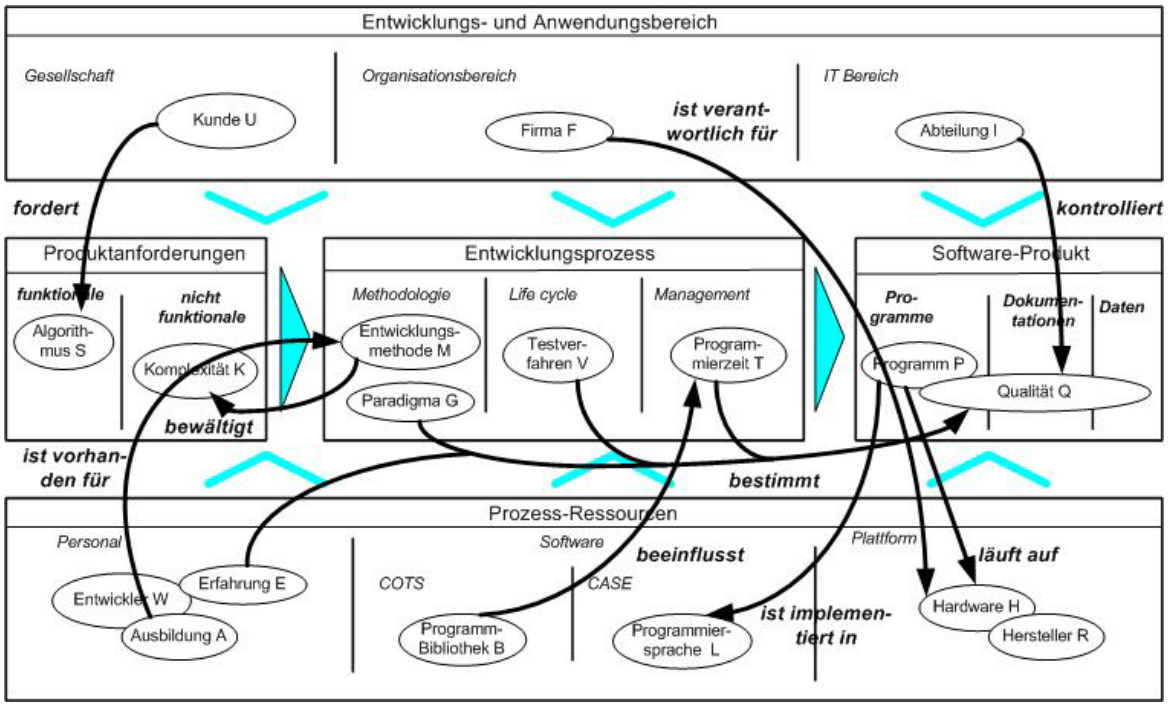


Abbildung 3: Mögliche kausale Zusammenhänge im Kontext der Software-Entwicklung

Sind diese Zusammenhänge schließlich durch Metriken beschrieben, so kann sich der folgende Artefakte-Bezug (hier wieder nur angedeutet) ergeben ([Abran 2010], [Kan 1995], [Putnam 2003], [Zuse 1998]). Die Metriken werden allerdings in der folgenden Abbildung nicht weiter erläutert und sind den Quellenangaben zu entnehmen.

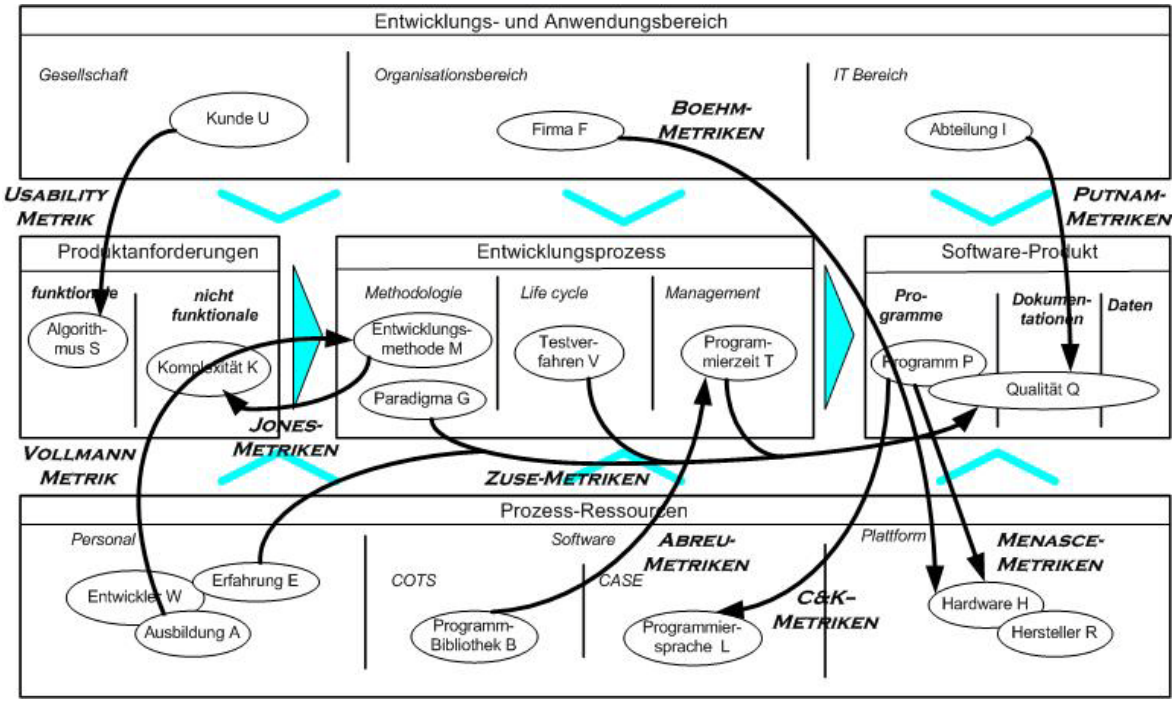


Abbildung 4: Mögliche metrikenbasierte Zusammenhänge im Kontext der Software-Entwicklung

Dabei können die ***Software-Metriken*** von einfachen Auszählungen bis hin zu ganzen, umfangreichen Bewertungskonzepten reichen. Beispiele (nach aufsteigender Komplexität geordnet) sind:

- *einfache Auszählungen*, wie Quellcodezeilen, Anzahlen von Modulen oder Klassen, Anzahl der Teammitglieder, Anzahl genutzter Programmiersprachen, Anzahl der Symbole einer Modellierungssprache usw. ([Kan 1995], [Zuse 1991]),

- *mathematische, metrische Skalen*, wie die Manhatten-Metrik, die Äquidistanz-Metrik und andere metrische bzw. projektive Skalen ([Fenton 1997], [Glaeser 2010]),

- *einfache Verhältniswerte*, wie die Defect Density (Anz(fehlerhafter Komponenten)/alle Komponenten) oder weitere Anteilsangaben, wie Kommentaranteil, Entwickleranteil usw., sowie Durchschnittsangaben, wie das Mean Time Between Failure (MTBF) oder auch MTFR, MTDD usw. oder die durchschnittliche Methodenanzahl einer Klasse ([Abran 2006], [Munson 2003]),

- *modellbezogene Anzahlen,* wie die zyklomatische Zahl und die Wohlstrukturiertheit von Programmflussgraphen oder der Kopplungsumfang und die Ebenenanzahl in Callgraphen ([Büren 2009], [Ejiogu 1991], [Zuse 1998]),

- *technologiebezogene Anzahlen*, wie die allgemein bekannten C&K-Metriken (Chidamber/Kemerer), die Abreu-Metriken oder die Strukturbewertungen von Simon für objektorientierten Programme, die kostenbezogenen Metriken für die Wiederverwendung von Poulin oder der COCOTS-Schätzansatz von Boehm für die komponenten-basierten Software-Systemen, die Service-Komplexitätsmetriken von Rud oder die klassischen QoS (Quality of Service) und SLA (Service Level Agreement) Bewertungen für die Service-orientierte Systementwicklung und schließlich die allgemeinen Entwicklungsmetriken von Wille oder die Trainings-Metriken von Russel und Performance-Metriken von Cortese für die agentenorientierte Software-Systementwicklung ([Boehm 2000], [Dumke 2010], [Lanza 2006], [Menasce 2002], [Poulin 2001], [Rud 2006], [Wille 2005]),

- *prozessbezogene Anzahlen*, wie die Bewertung der Anforderungsqualität nach der IEEE 982, der Entwicklungs-Personalaufwand nach Vollmann, die Architektur-Metriken nach Card, die Testmetriken von Liggesmeyer und Sneed oder die Wartungsmetriken von Lehner ([Card 1990], [Dumke 1992], [Lehner 1991], [Liggesmeyer 2002], [Sneed 2005], [Vollmann 1990]),

- *formelbasierte Berechnungen*, wie der durchschnittliche Programm-Dokumentationsanteil nach Thaller, die Programm-Lesbarkeit nach DeYoung und Kampens, die Entwicklungszeit nach der SLIM-Formel (Software Lifecycle Management) usw. ([Dumke 1994], [Putnam 2003], [Thaller 1994]),

- *quantitative Bewertungskonzepte*, wie die Aufwandschätzung nach dem COCOMO (Contructive Cost Model), die Software-Umfangsbestimmung nach dem COSMIC FP (Function Points), oder die Bestimmung der Prozessgüte nach dem CMMI (Capability Maturity Model Integration) mit den von Kulpa zusammen gestellten CMMI-Metriken ([Boehm 2000], [Dumke 2011a], [Kulpa 2003]).

Software-Metriken werden typischerweise im Zusammenhang mit so genannten Schwellwerten als Entscheidungshilfe eingesetzt. Andere Bewertungsformen können sich auf das Maxima oder Minima oder auf zulässige bzw. empfehlenswerte Intervalle beziehen.

Für die Anwendung der Metriken selbst ist die Skalenproblematik nach wie vor ein Kernproblem. Die Extrema reichen dabei von einer einfachen Benennung als ausreichendes Kriterium für eine "gut anwendbare" Messskala bis hin zur unbedingten empirisch sinnvollen Bewertung. Dabei reichen die Metriken-Anwendungen über multiple Metriken-Tupel bis hin zu der manchmal angestrebten *einen Kennzahl* für ein Bewertungskomplex. Typische mehrwertige Bewertungsprobleme bei der Anwendung mehrere Metriken für diese eine Kennzahl zeigt das folgende Beispiel. Man definiere beispielsweise die *Flugqualität* in folgender Weise:

Flugqualität = Durchschnittsgeschwindigkeit + Durchschnittsalter der Crew + Zufriedenheitsindex + durchschnittliche Kabinentemperatur in ^{o}C + Flugzeugtypklasse - Anz(Turbolenzen) - (Anz(Passagiere)/Anz(Plätze))

Hierbei sind alle qualitativen und quantitativen Skalenarten vertreten. Auch eine Normierung oder spezielle Wichtungen lösen diese Problematik nicht.

Andererseits gab und gibt es auch Effekte einer "Auszählung von allem Möglichen", wie beispielsweise die sehr hübsche Darstellung von Damrauer für den Alltag mit dem Beispiel [Damrauer 2008]

Aufzug = Elektrizität/Schwerkraft + Angst

Bei den ***Software-Maßen*** setzen wir eine Maßeinheit voraus und können hierbei zumeist sehr konkrete Entscheidungsunterstützungen erreichen. Beispiele hierfür sind:

- *gängige physikalische Maße,* wie Zeit als Antwort- und Reaktionszeit bei interaktiven Anwendungen und Warte-, Service- und Bearbeitungszeit bei Service-Lösungen ([Laird 2006], [Menasce 1998]),

- *ökonomische Maße,* wie Kosten, Personalaufwand und -zeit, ROI (Return on Investment), TCO (Total Cost of Ownership) usw. ([Poulin 2001], [Putnam 2003]),

- *soziologische Maße*, wie der IQ, die so genannte Stroud Number (als Fähigkeit gleichzeitig zu bearbeitender Prozesse), der U-Faktor (als durchschnittliche Unterbrechungszeit pro Tag) usw. ([DeMarco 1991], [Halstead 1976]),

- *hardwarebezogene Maße*, wie das Mooresche Gesetz (als Zeitmaß), die bereits oben genannten MTBF, MTFR, MTDD usw. und die Zuverlässigkeits- und Stabilitätsmaße ([Menasce 2002], [Peled 2001]),

- *softwarebezogene* Maße, wie $L_{Halstead}$ (als Halsteads Programmlänge), V_{McCabe} (als modifizierte McCabe-Zahl), KDSI (als Kilo Delivered Source Instruction, aus dem COCOMO), CFP (COSMIC Function Points), PM (als (Entwicklungs-) Personenmonate) usw. ([Abran 2010], [Dumke 2010], [Zuse 1998]).

All diese Maße implizieren eine Verhältnisskale und ermöglichen exakte empirische Vergleiche und damit auch Aussagen.

Die Anwendung von Software-Metriken und Software-Maßen geschieht auf der Grundlage von ***Messzielen*** (*Measurement Goals* $\mathcal{G}$). Diese sind durchaus unterschiedlich und reichen von einem (ersten) Verständnis bzw. Erlernen über ein Verbessern oder auch Zertifizieren bis hin zu einem für die Praxis relevanten Managen oder gar (als idealen Regelkreis) kontinuierlicher Qualitätssicherung. Eine qualitative Grundeigenschaft der Messziele ist im *SMART-Prinzip* (Specific, Measurable, Accountable, Realistic, Timely) charakterisiert [Ebert 2012]. Für die jeweiligen Ziele gibt es natürlich bereits umfangreiche Beispiele aber auch Systematiken und Vorgehensweisen von denen einige hier aufgelistet werden sollen:

- *Verstehen bzw. Untersuchen von Software*: Diese relativ "bescheidene" Zielsetzung kann jedoch für die weiteren, oftmals prozessbezogenen Zielstellungen von grundlegender Bedeutung sein. So impliziert beispielsweise die einfache Auszählung der Java-Bibliothek (Java 5 hat z. B. über 50000 Methoden) durchaus ein deutlicheres Komplexitätsverständnis für diese Technologie. Als Methoden hierfür können bereits die Vorstufe der Messung - die Modellierung - für die Identifizierung von Singularitäten bis hin zu geplanten und zielgerichtet durchgeführten Experimenten dienen. Eine detaillierte Anleitung zu Experimenten bzw. zahlreiche Beispiele finden sich in ([Basili 1986] und [Prechelt 2001].

- *Erklären bzw. Erlernen von Software-Merkmalen*: Ähnlich dem Verstehen werden hierbei allerdings noch weitere Zusammenhänge bzw. Prozesse in das Messziel involviert, die letztlich auch Aspekte der Umsetzung bzw. Anwendung der Messung beinhalten. Sehr hilfreich sind hierbei vor allem Visualisierungen von Strukturen und Eigenschaften bis hin zu 3D-Darstellungen (siehe [Lewerentz 2002] und [Zuse 1998]).

- *Verbessern von Software*: Die Verbesserung scheint ein immer aktuelles Messziel zu sein. Allerdings ist es dabei natürlich wichtig, die Ausgangsposition zu kennen. Methoden können dabei direkt auf das Software-Produkt gerichtet sein oder indirekt die Prozesse bzw. die eingesetzten Ressourcen aller Art betreffen (siehe [Dumke 2011], [Florac 1999], [Jones 2010] und [Witt 2012]).

- *Zertifizieren von Software*: Hierbei besteht das Messziel mit einer institutionellen bzw. rechtsträger-gebundenen Bestätigung für den Zustand, der im Allgemeinen vorgegebene Kriterien erfüllt (siehe [Bundschuh 2008], [Hollocker 1990] und [Kulpa 2003]).

- *Managen von Software*: Dies ist das eigentliche Messziel im Kontext einer realen Software-Entwicklung in der IT-Praxis. Auf den Prozess bezogen liegt hierbei eine Gütestufe 4 (nach dem CMMI) vor. Methoden hierfür sind in konkreten Regeln und Vorschriften fixiert und haben bezüglich der Auswertung klare Zuordnungs- und Aggregationsformen (Portfolios, Dashboards usw.) ([Ebert 2007], [Florac 1999], [Kenett 1999], [Pandian 2004]).

- *Steuern bzw. Gewährleisten von Software-Eigenschaften*: Dieses "höchste" Messziel impliziert einen nahezu geschlossenen Regelkreis, ist tool-basiert und liegt hauptsächlich für Aspekte der Qualitätssicherung von Software-Produkten in der Praxis vor ([Munson 2003], [Tayntor 2003]).

Natürlich sind die Messziele stets auf ***Messgegenstände*** (*Measurement Artefacts* $\mathcal{A}$) bezogen. Wir wählen diesen Begriff der Einfachheit halber, beziehen allerdings alle Merkmale, Komponenten und Prozesse einer Software-Entwicklung, -Wartung und Anwendung mit ein (wie wir sie beispielsweise oben bereits als Produkt, Prozesse und Ressourcen definiert haben). Beispiele für derartige Artefakte der unterschiedlichen Bereiche sind

- *Software-Produktartefakte:* funktionale Modelle, Architekturen, Implementationen, Bibliotheken, Datenbasen, Entwicklungs- und Anwenderdokumentationen, Zertifikate, Qualitätsreports usw. ([Abran 2006], [Kan 1995], [Liggesmeyer 2002], [Zuse 1998]),

- *Software-Prozessartefakte*: Entwicklungsphasen, Modellnotationen und -standards, Lifecycle-Modelle, Managementformen und -methoden (projekt- und qualitätsbezogen), Konfigurationen und Produktlinien, Wartungs-, Umstellungs- und Migrationsmethoden usw. ([Boehm 2007], [Jones 2010], [PMBOK 2004], [Royce 1998]),

- *Software-Ressourcenartefakte*: Entwicklungspersonal, CASE-Tools, COTS, Hardware-Ressourcen als Plattformen und Infrastrukturen usw. ([Büren 2009], [Goodman 1993], [Humphrey 2000]).

Diese Messartefakte können des weiteren noch andere Ausprägungen haben, die beispielsweise bestimmt werden

- *von den Systemarten:* wie z. B. Automotiv-Software, Informationssysteme, Wissenschaftlich-technische Berechnungssysteme, Verteilte Systeme, Kommunikations- und Entscheidungssysteme, Simulationssysteme usw. ([Ebert 1996], [Fenton 1997], [Sneed 2010]),

- *von den Software-Paradigmen*: das reicht von den klassischen prozeduralen über die objektorientieren und komponentenbasierten bis hin zu serviceorientierten und schließlich agentenbasierten System- und Entwicklungsformen. Eine mögliche Darstellung mit einigen angedeuteten Technologien, die wir hier nicht weitere erläutern wollen, zeigt die Abbildung 5. Andere Paradigmenklassifikationen sind beispielsweise die in formale und informelle bzw. in deklarative und imperative u. a. m. und bestimmen natürlich ebenso die jeweilige Ausprägung des Messartefaktes ([Dumke 1997], [Ebert 2007], [Georgieva 2010], [Rajlich 2006], [Schmietendorf 2002], [Schmietendorf 2010], [Sommerville 2010], [Wille 2005]),

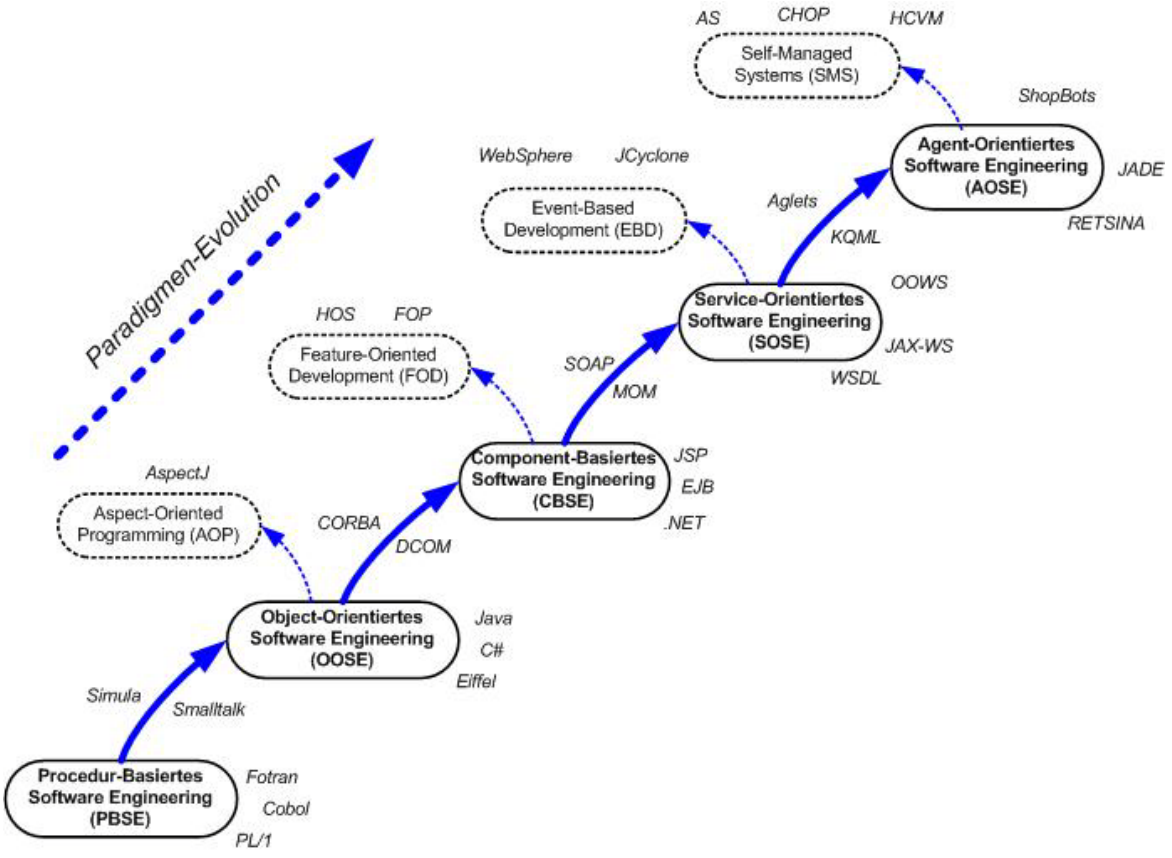


Abbildung 5: Technologiebasierte Klassifikation der Software-Paradigmen

- *von der Art der Integration*: als monolithische oder auch autarke, integrierte oder eingebettete, Teil- oder Folge-Komponenten bzw. Artefakte usw. ([Ejiogu 1991], [Sneed 2005], [Witt 2012]),

- *von der Prozessbezogenheit*: als Eingangs- oder finale, als Folge- oder separate, als Ausgangs- oder inkrementelle Komponenten bzw. Artefakte usw. ([Ezran 1998], [Keyes 2003], [McGarry 2002]) Diese Ausprägungsart der Software-Produkte bestimmt beispielsweise auch die Tauglichkeit der Anwendung von Metriken und deren Schwellwerte eines Paradigmas für das nächstfolgende [Dumke 2007].

Im speziellen Fall kann beispielsweise auch eine Messung an einem Pendant oder aber vergleichsweise ähnlichem Artefakt dem Messziel gerecht werden.

Die Anwendung von Software-Maßen und -Metriken wird durch die ***Messmethoden*** (*Measurement Methods* $\mathcal{M}$) vorgenommen. An dieser Stellen wollen wir allerdings noch einmal auf die Überschrift dieses Beitrages hinweisen. Sie schließt außer dem Messen auch das Bewerten mit all seinen Formen und Methoden mit ein, um möglichst stets eine Entscheidungsunterstützung zu gewährleisten. Die folgende Abbildung fasst einige Mess- und Bewertungsformen zusammen ([Bundschuh 2008], [Ebert 2007]).

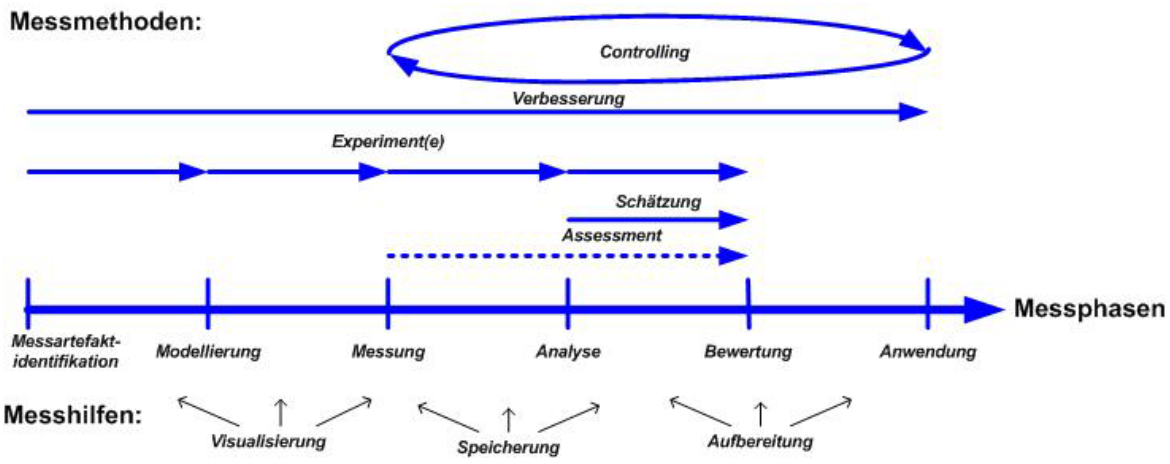


Abbildung 6: Software-Mess- und -Bewertungsmethoden

Diese Verfahren und Methoden zur Software-Messung und -Bewertung - kurz Messmethoden genannt - können wie folgt klassifiziert werden

Phasen der Software-Messung und -Bewertung:

- *Identifikation des Messartefaktes*: Dieser erste Schritt kann durchaus nichttrivial sein, wenn beispielsweise von einem Software-Effekt (Fehler, Ausnahme usw.) ausgegangen wird und der Messbereich und dessen Infrastruktur erst einmal identifiziert werden müssen ([Munson 2003], [Putnam 2003]).

- *Modellierung:* Die jeweiligen Messmethoden bzw. die Anwendung einzelner Metriken erfordert ohnehin eine Abstraktion der vorliegenden Artefakte (z. B. für Programme als Programmflussgraph, Datenflussgraph, Callgraph, Deskriptorgraph usw.) ([Card 1990], [Fenton 1997], [Nielson 1999]).

- *Messung:* Die Messung selbst ist natürlich der Kern einer Messmethode und erfordert die explizite Anwendung einer Software-Metrik bzw. eines -Maßes. Sie verläuft im Allgemeinen Tool-gestützt ist aber insbesondere bei Prozessmetriken häufig manuell bzw. durch Tools nur protokollierend unterstützt ([McGarry 2002], [Potter 2000], [Shepperd 1995], [SPC 1995]).

- *Analyse und Bewertung:* Hierbei geht es um die erklärungsgerechte Aufbereitung der Messdaten und deren zweckmäßige Einordnung in das domänspezifische Bewertungsumfeld ([Backhaus 1996], [Florac 1999], [Singpurwalla 1999]).

- *Anwendung:* Diese Phase ist eigentlich eine nachgestellte (wahlweise erscheinende) Aktivität, die allerdings unmittelbar eine Verbesserung bewirken kann oder sogar als Element eines Regelkreises für die ständige Qualitätssicherung dienen kann ([Jones 1991], [Kenett 1999]).

Spezielle Methoden der Software-Messung und -Bewertung:

- *Experiment bzw. Fallstudie:* Diese Form der Software-Messung kann alle oben genannten Messmethoden zur Anwendung bringen, dient allerdings einem speziellen Ziel und zwar der Untersuchung von Problemen der Software bzw. deren Entwicklung und Anwendung ([Ebert 1996], [Prechelt 2001], [Wohlin 2000]).

- *Assessment:* Auch beim Assessment können alle oben skizzierten Messmethoden zur Anwendung kommen. Allerdings handelt es sich hierbei um eine Zeitpunktbezogene und zumeist auch institutionell dokumentierte Messung und Bewertung, wie beispielsweise bei einer *Zertifizierung* ([Pfleeger 98]).

- *Evaluation:* Die Evaluation kann neben einer messbasierten Methode auch eigenständig (ohne Software-Messung) als reine empirische Bewertung angewandt werden. Es handelt sich hierbei z. B. um die fragebogenbasierten Erhebungen und anderen z. T. darauf aufbauenden statistischen Methoden der Trend-, Regressions- und anderen Analysen ([Bertold 2003, [Hanebutte 2003], [Pandian 2004]).

- *Schätzung:* Bei der Schätzung sind im Allgemeinen bereits Messungen erfolgt, die zu einer Berechnungs- bzw. Schätzformel führten und bei der Anwendung das Schätzen bzw. Bestimmen einiger, weniger Ausgangsgrößen erfordert ([Boehm 2000], [Jones 2007], [McConnel 2006], [Wille 2010]). Bei den Point-basierten Verfahren wird diese Schätzung allerdings nach Messvorschriften objektiviert (IFPUG Function Points, Use Case Points, Data Points, Object Points, COSMIC Function Points usw.) ([Bundschuh 2008], [Dumke 2011a], [Sneed 2005], [Symons 1991]).

- *Verbesserung:* Die separate Auflistung dieser Methode ist einem der populärsten Ziele gewidmet, der Verbesserung (Improvement), die die Anwendung der (aufbereiteten) Messergebnisse in einem vorgegebenen Kontext eben diese Verbesserung ausweist ([Emam 1998], [Kitchenham 1996], [McGarry 2002], [Lepasaar 2001]). Eine Möglichkeit der Bestimmung der Art bzw. des Niveaus der Verbesserung ist in [Dumke 2008a] beschrieben.

- *Managen:* Das hauptsächliche Ziel der Messung und Bewertung in einem praktischen Umfeld ist die quantitative Bestimmung der Produkt- und Prozesssituation und die darauf basierenden zweckgerichteten Entscheidungen (u. U. im Prozesslevel 4 nach dem CMMI) ([Balzert 2008], [Kulpa 2003], [Laird 2006], [Putnam 2003]). Beispiele für die bereits sehr umfangreichen Gütemodelle (MM steht hierbei stets für *Maturity Model*) sind *PEMM* (Performance Engineering MM), *TMM* (Testing Maturity Model), *SAMM* (Self-Assessment MM), *DMMM* (Data Management MM), *PMMM* (Project Management MM), *S^3MM* (Software Maintenance MM) u. v. a. m. (zu deren Details siehe [Braungarten 2005]).

- *Controlling:* Hierbei wird ein nahezu geschlossener Regelkreis angestrebt. Diese Form der Messmethodenanwendung wird zumeist nur bei der Programmbewertung und -qualitätssicherung erreicht ([Pandian 2004], [Wang 2000], [Witt 2012]). Beispiele für Akronyme derartiger Steuerungszyklen bzw. -prinzipien sind der Deming-Zyklus *PDCA* (Plan Do Check Act), der Messzyklus im Six Sigma Verfahren *DMAIC* (Define Measure Analyze Improve Control), der *EPPE-Prozess* nach der ISO 15939 (Establish, Plan, Perform, Evaluate) und das *E4* (Establish, Extract, Evaluate, Execute) ([Ebert 2007], [Meridji 2012], [Tayntor 2003]).

*Hilfsmethoden zur Software-Messung und -Bewertun*g:

- *Visualisierung:* Die Art der Visualisierung richtet sich natürlich nach dem Zweck bzw. den Kunden einer Software-Messung und Bewertung und reicht von den im Management-Bereichen beliebten Excel-Diagrammen bis hin zu speziellen Fokussierungen und schließlich 3D-Darstellungen ([Backhaus 1996], [Glaeser 2010], [Kan 1995], [Tetlow 2007]).

- *Transformation:* Hierbei handelt es sich vorrangig um die Anwendung statistischer Verfahren für die Aufbereitung der Messzahlen (Relevanz, Korrelation, Regressions- und Faktoranalyse, Skalentransformation usw.) ([Munson 2003], [Neumann 2010], [Singpurwalla 1999], [Zuse 1998]).

- *Simulation:* Neben der Simulation als eigenständige Bewertungsmethode kann diese auch als Hilfsmittel der Messdatenauswertung Anwendung finden [Pandian 2004].

- *Aggregation:* Die Aggregation unterschiedlicher Messwerte ist auch Bestandteil der zuvor genannten Transformation. Wir wollen sie hier nur separat in ihrer technologischen bzw. organisatorischen Einbindung beispielsweise als *Dashboards, Cockpits, Balanced Scorecards* bzw. *Portfolios* anführen ([Buglione 2003], [Dumke 2011], [Ebert 2007], [Heidrich 2009], [Liu 2008], [Neumann 2009]).

Abschließend zu den Messmethoden sei noch angemerkt, dass ähnlich wie beim Bohr'schen Atommodell, auch bei der Software-Messung ein Effekt der Messobjektveränderung auftreten kann, wie beispielsweise bei der kontinuierlichen Messung von Web Services (siehe [Rud 2006]).

Die erhaltenen ***Messzahlen*** (*Measurement Quantities* Q) aus der Anwendung mit Software-Metriken besitzen je nach Art der Metrikenberechnungsvorschrift einen ganzzahigen oder reellwertigen Wertebereich. Je nach Art der Software-Messung sind die erhaltenen Werte strukturiert, wie z. B. als Wertepaare, Tupel oder auch hierarchische Strukturen (beispielsweise bei der Function Point Messung) ([Juristo 2003], [Kitchenham 1996], [Sneed 2010]).

Innerhalb des IT-Entwicklungsprozesses werden derartige Messzahlen auch in Datenbasen implementiert und genutzt. Neben einfachen Metriken-Datenbanken [Bache 1994] werden auch komplexere Formen, wie Data Warehouses, Mediator- und Bus-Konzepte sowie servicebasierte Technologien angewandt (siehe [Braungarten 2005] und [Büren 2009]).

Die Anwendung des erhaltenen Metrikenwertes selbst ist allerdings nur unter Bezug einer speziellen Erfahrung möglich und zwar in Form so genannter Schwellwerte, der Orientierung auf maximale oder minimale Werte oder auch als Initialbildung für den künftigen Messwertvergleich. Dies führt uns zu einem weiteren grundlegenden Bestandteil der gesamten Software-Messung und -Bewertung.

Die ***Erfahrungen*** (*Measurement Experience* E) zur Software-Messung basieren auf bereits durchgeführte Messungen und Bewertungen und haben sehr unterschiedliche Formen und Inhalte. Wir wollen einige, insbesondere für die Software-Messung relevante davon hier kurz charakterisieren ([Boehm 1989], [Davis 1995], [Dumke 2007], [Endres 2003], [Kitchenham 2007], [Zelkowitz 2007]).

- *Schwellwerte*: Diese Erfahrung klassifiziert Metrikenzahlen hinsichtlich ihrer empirischen Bedeutung, beispielsweise als gut, (noch) zulässig oder problematisch.

- *Analogien*: Hierbei werden gültige empirische Vergleiche postuliert, die bei der Interpretation der Messzahlen Anwendung finden können.

- *Faustregeln*: Diese Form der Erfahrung kann ebenfalls für die Bewertung von Metrikenzahlen genutzt werden. Bei solchen Regeln, wie "1 Dollar Entwicklung kosten 2 Dollar Wartung", können Aussagen zu weiteren Software-Aspekten vorgenommen werden.

- *Korrelationen*: Diese Erfahrung hilft bereits bei der Auswahl der Metriken für ein Messvorhaben. Aber Achtung: Im Allgemeinen korrelieren die Lines of Codes mit der McCabe-Zahl für ausgewählte Programme. Allerdings haben beide Metriken eine völlig unterschiedliche empirische Bedeutung.

- *Gesetze:* Bei dieser Erfahrung liegt bereits eine verallgemeinerte Darstellung vor. Sie kann anstelle einer Software-Messung oder zur Bestimmung der Gültigkeit dieser Erfahrung für andere/neue Artefakte Anwendung finden.

Die Problematik derartiger Erfahrungen liegt allerdings darin, dass sie zumeist zu sehr generalisiert sind und die Produkt- und Prozessspezifika kaum mehr erkennen lassen, so dass sie häufig nur noch Orientierungscharakter besitzen. Die dennoch vorhandene besondere Bedeutung dieser Erfahrungen für ein Unternehmen führte zum Konzept der so genannten ***Experience Factories***, die insbesondere firmenspezifisch relevant sind ([Basili 2007], [Ciolkowski 2002]).

Die gewonnenen ***Maßzahlen*** (*Measurement Values* V) aus der Anwendung von Software-Maßen können unmittelbar zur Bewertung verwendet werden. Diese Maßzahlen haben natürlich auch ***Maßeinheiten*** (*Measurement Units* U), die wir bereits oben charakterisiert haben. Sie besitzen daher in der Regel eine Verhältnisskala und ermöglichen die "gewohnten" Anwendungen statistischer Auswertungen. Auch diese Zahlen werden zumeist in Datenbasen organisationsspezifisch strukturiert, um eine zweckmäßige Auswertung zu ermöglichen. Einfachste Strukturierungsformen sind Messwertreihen für prozessbezogene Auswertungen oder Messtupel für die beispielsweise mehrdimensionale Qualitätsbewertung ([Boehm 2007], [Fenton 1997], [Jones 2010], [McGarry 2002], [Zuse 1998]).

Eine besondere Erfassungsform und -bereitstellung von Projektdaten ist in der Datenbasis der so genannten ***ISBSG*** (International Software Benchmark Standard Group) gegeben. Auf Freiwilligenbasis sind hier bereits Messdaten von über 5000 Projekten gespeichert und für geringe Kosten für den eigenen Bereich nutzbar. Neben Produkt- und Prozessbezogene Daten werden hier auch Aufwandsdaten gespeichert, die eine sinnvolle Anwendung für den eigenen Bereich ermöglichen (siehe auch [Ebert 2007] und [Wille 2010]). An dieser Stelle soll noch darauf hingewiesen werden, dass gängige Praxis die gleichzeitige Anwendung von Software-Metriken und Software-Maßen darstellt.

Generell können wir auch an dieser Stelle konstatieren, dass die Softwaremessung und -Bewertung zumeist zu ***neuen Erfahrungen*** (*Extended Measurement Experience* E') und in seiner Anwendung auch zu ***verbesserten Messgegenständen*** (*Improved Measurement Artefacts* A') selbst führt.

Für die Software-Messung werden selbstverständlich ***Mess-Tools*** (*Measurement Tools* T) eingesetzt (der Nutzung von Mess- *und* Bewertungsmethoden entsprechend auch CAME-Tools (Computer Assisted Measurement and Evaluation) genannt). Für die Mess-Tools gibt es unterschiedliche Klassifikationsformen, die je nach ihrer Einordnung in die IT-Prozesse auch unterschiedliche Bereiche der Software-Messung und -Be-wertung abdecken. Wir wollen hier einige dieser Klassifikationen kurz charakterisieren.

- *Mess-Tools zu den Messphasen:* Nach dieser Klassifikation unterschieden wir grob in Modellierungs-, Mess- und Analyse- bzw. Auswertungs-Tools. Dabei können eine oder mehrere Messphasen in einem Tool integriert sein. So ein klassisches Mess-Tool, wie Logiscope vereint die Modellierungsformen mit der statischen und dynamischen Programmmessung ([Abran 2006], [Dumke 1996]). Auch eigenständige statistische Softwarepakete kommen dabei insbesondere für statische Analysen zum Einsatz (siehe [Backhaus 2010] und [Juristo 2003]).

- *Mess-Tools für die Entwicklungsphasen:* Ähnlich wie die Upper- und Lower-CASE-Tools gibt es hierbei Messtools für die Anforderungsanalyse, die Spezifikation (z. B. für UML oder formale Spezifikationssprachen), für den Design und vor allem aber für die Implementation, die vor allem die statischen Merkmale aber auch die Testqualität und Performanz bewerten. Dabei sind teilweise Phasenbewertungen zusammengefasst und ermöglichen insbesondere den Vergleich von Spezifikations- und Implementationseigenschaften ([Munson 2003], [Ebert 2004], [Sneed 2010], [Whitty 1990]).

- *Mess-Tools für die jeweiligen Messmethoden*: Nach der obigen Klassifikation sind dies vor allem Tools für die Aufwandschätzung (FP Workbench, SLIM, COCOMO, COSMICXpert usw.), für die Artefact-Messung und Bewertung (Logiscope, SMR, FindBugs, Bugzilla, Sotograph usw.) und für das Management bzw. Projekt-Controlling (ARIS, Hackystat, KnowledgePlan, Seer, CostXpert, Eclipse Process Composer, ISBSG, PEMM usw.) ([Boehm 2000], [Bundschuh 2008], [Ebert 2011], [Frohnhoff 2009], [Pandian 2004], [Sneed 2005]).

- *Mess-Tools und Vorgehensmodelle:* Hierbei berücksichtigen die jeweiligen Mess-Tools die Art der Vorgehensmodelle, wie evolutionäre Entwicklung, V-Modell, agile Entwicklung usw., oder aber auch den jeweiligen Anwendungsbereich, wie der Automotive-Bereich, der Banken-Sektor, der öffentlich-rechtliche, administrative Bereich usw. Mess-Tool-Beispiele hierfür sind daher (EPF Composer, SOFT-AUDIT, Rational Method Composer, MeterIT, UnitMetrics usw.) ([Ebert 2011], [Garcia 2011], [Jones 1991], [Putnam 2003], [Schmietendorf 2012], [Sneed 2005]). Man beachte allerdings, dass gerade für den Bereich der Vorgehensmodelle die Mess-Tools zumeist einen integrativen Bestandteil der Entwicklungs-Tools darstellen [Yazbek 2010].

- *Mess-Tools für die Software-Paradigmen:* Gemäß der oben angegebenen Klassifikationsform für Software-Paradigmen unterscheiden wir auch die Mess-Tools für das OOSE (SDMetrics, SMTCPP, CrocoCosmos, Logiscope usw.), für das CBSE (EJB Measurement, COCOTS, COMET usw.), für das SOSE (BpelMeter, ITIL, ActiveBPEL, ARM usw.) und für das AOSE (MUMMS, MAS FP, AgentBench, PerfAglet usw.) ([Boehm 2000], [Dumke 1997], [Ebert 2007], [Menasce 2002], [Poulin 2001], [Rud 2009], [Schmietendorf 2007], [Schmietendorf 2002], [Simon 2001], [Welke 2011], [Wille 2005]).

- *Technologiebasierte Mess-Tools:* Die jeweils für das Mess-Tool verwendete Technologie hängt von der Art bzw. Ausrichtung dieses Mess-Tools ab. Es kann eine einfachste *Erweiterung eines CASE-Tools* sein (z. B. als Eclipse Plugin), oder eine CASE-Anbindung als *eigenständige Komponente* (z. B. das MetricsOne für die Rational-CASE-Tools oder aber auch eine eigenständiges Mess-Tool oder gar *Mess-Infrastruktur* mit einem Artefakt-Input, eigenständigen Mess- und Auswertungsfunktionen einschließlich einer mehr oder weniger umfangreichen Messdatenbasis. Darüber hinaus kommen natürlich auch aktuelle bzw. moderne Technologien zur Anwendung, wie die Service-Orientierung, die zu Mess- und Auswertungsangeboten im Web führt (bekannt als *e-Measurement* oder auch *ubiquitous Measurement* mit den Beispielen des ViSEK-Service für den KMU-Bereich, des Java-Messservice OOMJ und der Messdienste im SML@b) oder auch die Agententechnologie für die zweckmäßige Auswahl derartiger Dienste (siehe die OWL-Implementation der ISO 15939 im System Artemis) ([Ciolkowski 2002], [Farooq 2005], [Kunz 2010], [Lother 2007], [SML@b 2012]). Die spezielle Ausmessung des Web als Messgegenstand selbst hat sich als *Web Tomographie* etabliert [Dumke 2004].

- *Qualität der Mess-Tools:* Hierbei handelt es sich nicht explizit um eine Klassifikation, sondern um die besonderen Eigenschaften der Mess-Tools, die allerdings auch zu entsprechenden Klassifikation führen können. Als typische Eigenschaften haben sich dabei unter anderem erwiesen ([Dumke 1996], [Yazbek 2010], [Zuse 1998]):

  Stabilität eines Mess-Tools: Diese Eigenschaft weist darauf hin, dass die Software-Messung die jeweilige Technologie bzw. Implementationsform des Messartefaktes korrekt berücksichtigt und stets dieselben zuverlässigen Messwerte für gleiche Eigenschaften erzeugt. Das bedeutet beispielsweise für eine Quellcodemessung, dass alle Eigenschaften der Codeerzeugung (des jeweiligen Compilers) berücksichtigt werden, was durchaus nichttrivial ist.

  Validität eines Mess-Tools: Hierbei geht es um die korrekte Umsetzung bzw. Erzeugung des Metriken- bzw. Messwertes gemäß der Messwertdefinition. Probleme hierbei können aufgrund einer falschen, unterschiedlichen Interpretation oder einer unzureichenden bzw. relativ "offenen" Definition der Metriken (wie z. B. bei der WMC- und der LCOM-Metrik der so genannten C&K-Metriken) entstehen {Whitmire 1997].

  Flexibilität eines Mess-Tools: Diese Eigenschaft betrifft vor allem Möglichkeit, auf der Grundlage vorgegebener Modellierungs- und Messformen das Tool um eigene Metriken zu ergänzen. So besitzt beispielsweise das COSMOS-Tool eine eigene *Metrikendefinitionssprache*, die eine eigene Komposition von Programmgraphelementen für neue, eigene Metriken ermöglicht [Fenton 1997].

  Offenheit eines Mess-Tools: Hierbei geht es um die Möglichkeit, nur Teile eines Mess-Tools verwenden zu können. Also beispielsweise die Messdaten zu übernehmen und eine eigene Messdatenauswertungsform anzuwenden.

  Untersuchungen haben ergeben, dass insbesondere bei eigenständigen Mess-Tools die Eigenschaften der Flexibilität und Offenheit zum Teil nur wenig ausgeprägt sind ([Abran 2006], [Dumke 2008]).

Letztendlich wird zur Software-Messung natürlich auch ein ***Messpersonal*** (*Measurement Personal* $\boldsymbol{P}$) benötigt. Hinsichtlich der in der Software-Messung und -Bewertung involvierten Perdonals unterschieden wir in ([Dumke 2003], [Goodman 1993], [Pandian 2004], [Pfleeger 1998], [Royce 1998]).

- *explizites Messpersonal*: Hierzu zählt das Personal, welches unmittelbar mit der Software-Messung und -Bewertung als Hauptgegenstand ihrer Arbeit zu tun hat, wie Analytiker, Metriken-Bibliothekar, Metriken-Aufbereiter, Zertifizierer, Messplaner und Leiter, QA-Personal, Experimentator, Metriken- oder Messstandardentwickler usw.

- *implizites Messpersonal*: Diese Bezeichnung wollen wir dem Personal geben, welches die Software-Messung "nur" nutzt und zum üblichen IT-Personal zählt, wie der Systemanalytiker, Entwickler, Akquisiteur, Designer, Programmierer, Tester, Wartungsingenieur usw.

Bezüglich der Eigenschaften bzw. Fähigkeiten können wir für das Messpersonal folgende Merkmale konstatieren ([Büren 2009], [DeMarco 1991], [Humphrey 2000], [Richter 20012], [SPC 1995], [Witt 2012]).

- *Qualifikation:* Diese Eigenschaft erscheint selbstverständlich, weist aber auf die Notwendigkeit hin, alle Ausbildungsformen des IT-Personals auch wirklich in dieser Hinsicht auszurichten.

- *Motivation:* Diese Eigenschaft ist unbedingte Voraussetzung für die Berücksichtigung der Software-Messung überhaupt. Sie impliziert auch die ISO 9000-Regel "Einbeziehung bzw. Verantwortung des Top-Managements für das quantitative Projektmanagement" für die Chance einer Implementation von Software-Messprozessen.

- *Engagement*: Nach unseren Erfahrungen wird diese Eigenschaft häufig unterschätzt. Genau darum orientieren sich viele Untersuchungen darauf, den Aufwand für die Installation und vor allem für eine erfolgreiche Weiterführung von Messprozessen zu bestimmen und so entsprechend auch einhalten zu können. Ein explizites Personal für die Software-Messung und -Bewertung ist in der Regel unumgänglich.

In diese Rahmen fallen auch die Betrachtungen zur Unabhängigkeit des Messpersonals bezüglich der Gewährleistung objektiver Ergebnisse ([Buglione 2012], [McConnel 2006], [Sneed 2005]).

Im Kontext des Messpersonals gehört unbedingt die Betrachtung der ***Communities***. Für den Bereich der Software-Messung und -Bewertung wollen wir hier die folgenden Communities anführen ([Bundschuh 2008], [Dumke 2003], [Ebert 2007], [SML@b 2012]).

- *Berufsverbände:* Im deutschsprachigen Raum gibt es hier natürlich die *Fachgruppe 2.1.10 der GI* (Gesellschaft für Informatik) und den *DASMA-Verein* (Deutsche Anwendergruppe für Software-Metrik und Aufwandschätzung). International seien hierbei die *COSMIC-Community* (Common Software Measurement International Consortium), die *MAIN* (Measurement Associations International Network), die *CECMG* (Central Europe Computer Measurement Group) sowie die nationalen Function-Point-Gruppen (die *NESMA* in den Niederlanden, die *FISMA* in Finnland, die *ISMA* in Italien usw.) genannt [Bundschuh 2008].

- *Institutionen:* Neben den internationalen Einrichtungen, wie das *SEI* (Software Engineering Institute) in Pittsburgh, das *SEL* (Software Engineering Lab) in Maryland und das *ESI* (European Software Institute in Bilbao) sind in Deutschland vor allem das *IESE* (Fraunhofer Institut für empirisches Software Engineering) in Kaiserslautern, die Unis in Cottbus, Kaiserslautern, Magdeburg und Paderborn sowie die Fachhochschulen in Berlin (HWR), Bingen, Köln und Stralsund zu nennen [Ebert 2007].

- *Fachliteratur:* Neben den einschlägigen hier zum Teil bereits aufgeführten Bücher sind das die Journals 'Empirical Software Engineering', 'Software and Systems Engineering', 'Software Quality Journal', unsere 'Software Measurement News' u. v. a. m. Nicht zuletzt seien auch die vielen Web-Angebote, vom Wikipedia, den Tool-Herstellern, den Community-Angeboten bis hin zum SML@b genannt.

- *Konferenzen*: Explizite Metriken- bzw. Software-Messkonferenzen sind beispielsweise die *EASE* (International Conference on Empirical Assessment in Software Engineering), die *ESEM* (Empirical Software Engineering), die *IFPUG*-Konferenzen der International Function Point User Group, die *STAREAST* (Software Testing Analysis & Review Conference), die *SMEF* (Software Measurement European Forum) in Rom, die *IWSM/Mensura* (International Workshop on Software Measurement), sowie im deutschsprachigen Raum die *SWQD* (Software Quality Days) in Wien, die *ASQT* (Arbeitskonferenz Softwarequalität und Test) in Klagenfurt, die *CONQUEST* (International Conference on Software Quality) in Nürnberg, die *BSOA* (Bewertung service-orientierter Architekturen) und die *MetriKon* (Metriken Konferenz) und können unter [SML@b 2012] aktuell eingesehen werden.

Diese kurze Auflistung wesentlicher Inhalte bzw. involvierter Komponenten bei der Software-Messung definieren insgesamt ein ***Software-Messystem*** (Measurement System) [Skyttner 2005]

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

mit den Messsystemkomponenten M_{MS} und den Relationen R_{MS} zwischen diesen Komponenten. Die prozessbezogene Sicht führt uns schließlich zum ***Software-Messprozess*** SMP. Eine vereinfachte Darstellung dieses Messprozesses einschließlich seiner Anwendung als idealisierten Regelkreis zeigt die folgende Abbildung.

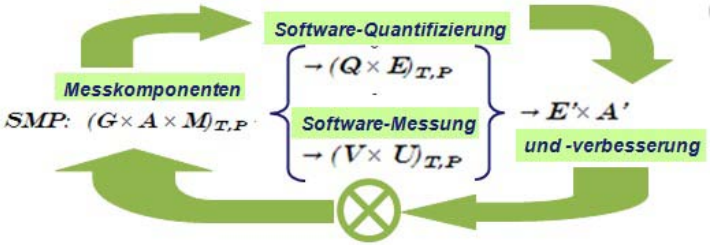


Abbildung 7: Vereinfachte Formalisierung des Software-Messprozesses

Durch weitere Indexierung und damit Spezifizierung können somit spezielle Messprozesse prinzipiell, allerdings auch nur sehr grob charakterisiert werden, wie beispielsweise ([Georgieva 2009], [Jones 1991], [Laird 2006]):

- *eine einfachste/erste Metrikenanwendung*: $SMP_{\textbf{produktbezogen}}$:

$$(G_{Bewertung} \times A_{Software\text{-}Produkt} \times M_{Messung})_{T_{CAME\text{-}Tool}, P_{QA-Ingenieur}} \rightarrow (Q_{normalisiert} \times E_{Schwellwerte})$$

- *eine projektbezogene Aufwandschätzung*: $SMP_{\textbf{prozessbezogen}}$:

$$(G_{Managment} \times A_{Software\text{-}Produkt} \times M_{Schätzung})_{T_{CAME\text{-}Tool}, P_{ZertifizierterSchätzer}} \rightarrow (V_{umfangsbasiert} \times U_{Personenmonate})$$

- *eine Prozessverbesserung*: $SMP_{\textbf{prozessbezogen}}$:

$$(G_{Verbesserung} \times A_{Software\text{-}Prozess} \times M_{Assessment})_{T_{CASE\text{-}Tool}, P_{Projektleiter}} \rightarrow (Q_{aggregiert} \times E_{Prozesslevel})$$

Durch die Verwendung einer erweiterten Indexierung für die Charakterisierung der Komponenten eines Messsystems, etwa in der Form (ohne dies her weiter zu vertiefen)

$$^{Prozess}_{Evidenz}\ MESSKOMPONENTE\ ^{Paradigma}_{Art/Form/Ausprägung}$$

ist es möglich das ***Software-Messniveau*** (Software Measurement Level) zumindest ordinal zu bewerten und somit eine deklarative Vergleichsform zu gewährleisten [Dumke 2011]. Hilfreich ist diese Bewertungsform auch für die Identifizierung offener bzw. unzulänglicher Metrikenanwendungen in den verschiedenen softwaretechnischen Bereichen.

Software Measurement Frameworks

Eine spezielle Form der Charakterisierung von Messprozessen, -methoden und -modellen wird in Form von so genannten ***Software Measurement Frameworks*** *SMF* charakterisiert. Dabei gibt es unterschiedliche Klassifikationen, die sich auf die Beschreibungsform (*operational* oder auch (nur) *deklarativ*), auf die Art der Technologieanwendung (*arbeitsplatzbezogen* oder *infrastrukturell*) oder auch auf die Ausrichtung bzw. Berücksichtigung aller Aspekte einer Messprozesskomponente (als *speziell* bzw. *aspektbezogen* oder *ganzheitlich*) beziehen. Allgemein können wir diese Frameworks als

$$SMF = \{SMP,\ ORG\}$$

charakterisieren, wobei ORG die Aktivitäten und Maßnahmen zur Einführung, Initialisierung und Gewährleistung eines Software-Messprozesses beschreibt. Bei der konkreten Beschreibung von Measurement Frameworks wollen wir uns im Folgenden auf die jeweilige Ausprägung der Messprozesskomponenten beschränken. Wir wollen diese Beschreibung allerdings sehr kurz halten und verweisen dabei unbedingt auf die angegebene Literatur.

Generell ist dabei allerdings zu berücksichtigen, dass diese Messprozessanwendungen stets in einem Software-Prozess (Entwicklung, Wartung und Anwendung) zu integrieren sind bzw. integraler

Bestandteil sein sollten und nach unserer Meinung natürlich auch müssten (siehe [Balzert 2008], [Boehm 2007], [Emam 1998], [Kulpa 2003], [PMBOK 2004], [Putnam 2003], [Sommerville 2010] und [Witt 2012]).

- *deklarative Software Measurement Frameworks*: Das ***Measurement Framework von Zuse*** $\mathcal{SMF}_{Zuse}$ basiert auf der Messtheorie [Zuse 1998]. Einige der wesentlichen Inhalte sind

  $requirementsForMeasures = \{req_{general}, req_{rankingProperties}, req_{units}, req_{substitutionOperation}, req_{additiveRatioScale}, req_{non\text{-}additive}, req_{densityMeasures}, req_{wholeness}, req_{predictionModels}, req_{sizeMeasures}, req_{simpleMeasures}, req_{componentIndependence}\}$

  $typesOfMeasures = \{TM\text{-}CNT, TM\text{-}ADD, TM\text{-}HYA, TM\text{-}HYAS, TM\text{-}DEN, TM\text{-}DENM, TM\text{-}NADDR, TM\text{-}BSA, TM\text{-}HYM, TM\text{-}ORD, TM\text{-}PC, TM\text{-}FB, TM\text{-}MIMA, TM\text{-}IND, TM\text{-}NGEXT\}$

  wobei gilt *TM-CNT* (counting on attribute), *TM-ADD* (additive measure), *TM-HYA* (hybrid measure by additivity), *TM-HYAS* (hybrid measure additive plus a constant), *TM-DEN* (density measures), *TM-DENM* (density measure with an additional condition), *TM-NADDR* (non-additive ratio scale), *TM-BSA* (two concatenation operations), *TM-HYM* (hybrid measure by multiplication), *TM-ORD* (purely ordinal measure), *TM-PC* (percentage measure), *TM-FB* (modified function of belief), *TM-MIMA* (minimum – maximum), *TM-IND* (independence conditions) und *TM-NGEXT* (negative extension structure and non-additive ratio scale). In diesem Framework werden folgende Messartefakte betrachtet

  $\mathcal{A}_{Framework} = \{\mathcal{A}_{ISO9000\text{-}3}, \mathcal{A}_{FunctionPoints}, \mathcal{A}_{COCOMO}, \mathcal{A}_{maintainability}, \mathcal{A}_{structureCharts}, \mathcal{A}_{informationFlow}, \mathcal{A}_{measuresOfBowles}, \mathcal{A}_{coding}, \mathcal{A}_{testing}, \mathcal{A}_{cohesion}, \mathcal{A}_{maintenance}, \mathcal{A}_{documentQuality}, \mathcal{A}_{OOmeasures}, \mathcal{A}_{lifeCycle}\}$

  Die Skalenanalyse wird auf der Basis folgender Operationen für extensive Strukturen vorgenommen

  $\mathcal{M}_{modelling} \in \{BALT, BSEQ, CINT, CUNI, DSEQ, HAGG, HGEN, LSEQ, RALT, RSEQ, SBSEQ\}$

  wobei gilt *BALT* (parallel concatenation), *BSEQ* (sequential concatenation), *CINT* (object-oriented concatenation), *CUNI* (object-oriented unification), *DSEQ* (charts sequential concatenation), *HAGG* (object-oriented hierarchical concatenation), *HGEN* (object-oriented generalizational concatenation), *LSEQ* (boards sequential concatenation), *RALT* (resistors parallel concatenation), *RSEQ* (resistors sequential concatenation) und *SBSEQ* (source code sequential concatenation). Damit ist eine eindeutige Klassifikation der Software-Metriken- und -Maßeigenschaften möglich und für Vergleiche anwendbar.

  Als ein weiteres deklaratives Measurement Framework ist der ***ISO 15939 Standard*** $\mathcal{SMF}_{ISO15939}$ [ISO 15939 2001]. Das folgende Schema kennzeichnet die jeweiligen Bereiche bzw. Komponenten dieses Standards.

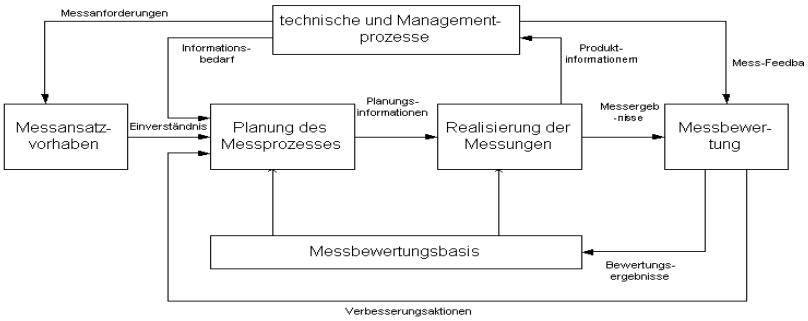


Abbildung 8: Grundkomponenten der ISO 15939

Dieser bereits oben erwähnte Standard gibt gewissermaßen das Gerüst eines Messprozesses vor. Der besondere Wert besteht in der Identifikation des "Messansatzvorhabens", der die Software-Messung als Kern einer Qualitätssicherung als *Option*, wie es eben auch noch zum Teil Praxis ist, identifiziert. Zum anderen bedeutet die "Messbewertung" nicht etwa die Auswertung der Messdaten, sondern der Nachweis der Tauglichkeit der angewandten Metriken bzw. des Messansatzes überhaupt [McGarry 2002]. Auch das ist nicht immer gewährleistet (s. u.).

- *operationales Software Measurement Framework*: Als Measurement Framework dieser Art kann die ***GQM-Methode*** (Goal Question Metric) SMF_{GQM} angesehen werden ([Basili 1994], [Solingen 1999]). Die Operationalität besteht hier in der (genauen) Reihenfolge der Anwendungsschritte in einem ganzheitlichen Kontext. GQM bedeutet dabei allerdings nicht nur die Planung der Software-Messung (als Ziele, Fragen bzw. Hypothesen und deren Beantwortung durch anzuwendende Metriken), sondern schließt die Messung selbst und deren Auswertung bezüglich der Beantwortung der Fragen ein. Die GQM-Erweiterung als $GQM^{+Strategy}$ bedeutet darüber hinaus, dass die Zielbildung selbst in eine Business-Strategie eingebettet und damit eingeordnet wird, wie in der folgenden Abbildung kurz skizziert ist [Basili 2007a].

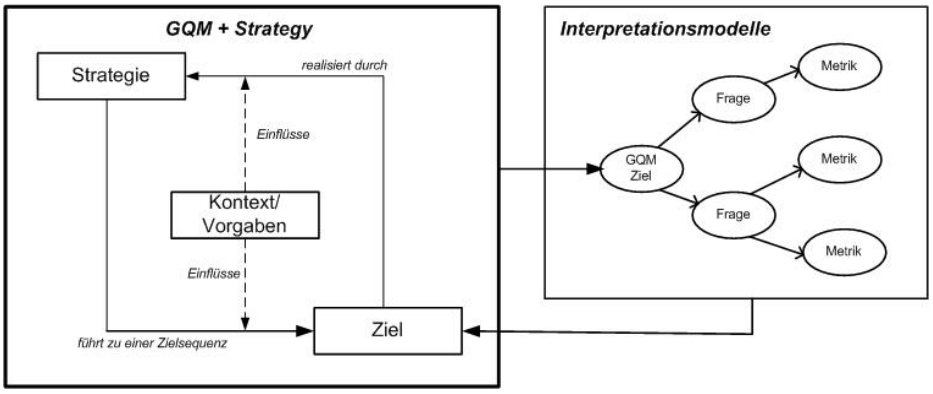


Abbildung 9: GQM-Erweiterung zum $GQM^{+Strategy}$

Als weiteres operationales Measurement Framework (nicht nur auf die Software-Entwicklung bezogen) kann der so genannte ***Six-Sigma-Ansatz*** $SMF_{SixSigma}$ angesehen werden [Tayntor 2003]. Dabei geht um eine nahezu "Nullfehlerorientierung" und die darauf beruhenden Schritte bzw. Maßnahmen [Fehlmann 2010].

- *infrastrukturelle Measurement Frameworks*: Wie oben bereits erwähnt, kann natürlich auch das Internet selbst zur Unterstützung der Software-Messung und -Bewertung eingesetzt werden. Einen umfassenden Ansatz zur Implementation der ISO 15939 auf der Grundlage von Web Service ist in [Kunz 2010] beschrieben als SMF_{SOA}. Grundlage für diese und ähnliche Ansätze sind ***Messontologien.*** Eine derartige Ontologie für die Objektorientierung ist in der Abbildung 10 angegeben [Wang 2001]. Inzwischen nutzt man derartige Konzeptdarstellungen auch für andere Paradigmen, um möglichst alle Komponenten eines Software-Messsystems zu berücksichtigen. Die oben skizzierte SOA-basierte Lösung wird bereits in der Praxis für die Java-basierte Systementwicklung erfolgreich eingesetzt [Dumke 2009].

- *aspektbezogene Measurement Frameworks:* Bei diesen Measurement Frameworks liegt eine gewisse Fokussierung bzw. Spezialsierung vor - also eine Aspektbezogenheit. Als Erstes wollen wir hier den Aspekt der Verbesserung nennen, wie er sich vor allem in den Prozessverbesserungskonzepten und Bewertungsmodellen der CMMI, dem SPICE oder dem ITIL (IT Infrastructre Library) manifestiert, kurz $SMF_{Improvement}$ genannt ([Emam 1998], [Johnson 2007], [Kulpa 2003]) mehr oder weniger explizit darstellt..

  Eine weitere Fokussierung ist die (korrekte) Anwendung statistischer Methoden insbesondere für die Prozesssteuerung, kurz SMF_{SPC} genannt ([Florac 1999], [Kitchenham 1996], [Pandian 2004]).

Schließlich, insbesondere unter besonderen technologischen Methoden und Verfahren seien hier die adaptiven Measurement Frameworks $SMF_{adaptiv}$ und die proaktiiven Measurement Frameworks $SMF_{proaktiv}$ genannt, die eine situative und sogar eigenständige Einflussnahmen bzw. Messauslösung und -kontrolle initiieren ([Mencke 2008], [Neumann 2009], [Wille 2005]).

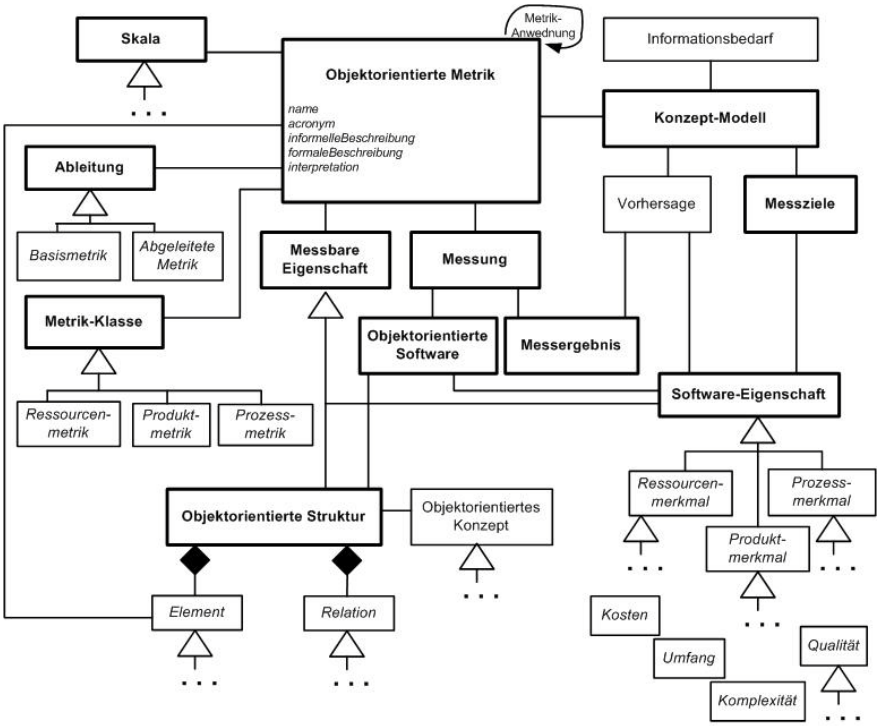


Abbildung 10: Messontologie für die Messung objektorientierter Systeme nach Wang

- *ganzheitliche Messansätze (Holistic Measurement Frameworks):* Bei der Ganzheitlichkeit beziehen wir uns auf die möglichst vollständige Berücksichtigung der Messziele und/oder der Messartefakte und/oder der Messmethoden. Die Vollständigkeit der Messziele bedeutet:, dass hierbei die wesentlichen Organisationsziele bzw. Business-Zielstellungen grundsätzlich einbezogen sind. Ein ganzheitlicher Ansatz dieser Art ist der E4-Ansatz SMF_{E4} [Ebert 2007]. Die Grundidee ist die direkte Einbettung in die Business-Prozesse und deren systematische, messbasierte Gewährleistung. Die folgende Abbildung zeigt derartige Messinhalte für die Business-Zielerreichung aus [Ebert 2007]. Eine anderes, weiteres ganzheitliches Measurement Framework dieser Art ist das $SMF_{GQM+Startegy}$.

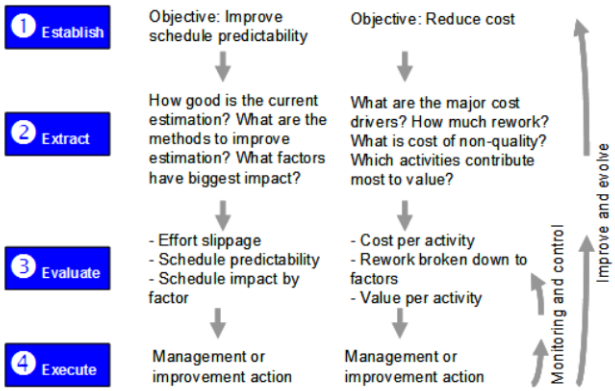


Abbildung 11: E4-Messformen von den Business-Zielen abgeleitet

Die Vollständigkeit und damit Ganzheitlichkeit des Messansatzes bezüglich der Messartefakte wird durch das *CAME* (Choice, Adjustment, Migration, Efficiency) Measurement Framework SMF_{CAME} angestrebt. Dabei bestimmt das *Choice* die Auswahl des Messbereichs aus allen möglichen Artefakten des Software-Produktes, des Software-Prozesses und der dafür notwendigen Software-Ressourcen. Das *Adjustment* steht für die Bestimmung der Skaleneigenschaften der ausgewählten Metriken oder gar Maße. Die *Migration* steht für die Auswahl und Installation bzw. Integration des jeweiligen Messprozesses bzw. der Messform und das *Efficiency* charakterisiert die Art bzw. den Umfang der Tool-Unterstützung durch die so genannten CAME-Tools. Zur Unterstützung der Operationalität dieses Ansatzes wurden Software-Mess-Kategorien definiert ([Awodey 2010], [Neumann 2012a], [Ormandjieva 2010]).

Schließlich wollen wir aus der Sicht eine vollständigen Berücksichtigung der Messmethode, d. h. diese Methode ganzheitlich bezüglich Methode, Konzept, Kontext, Tool-Unterstützung und Mess-Tool-Quallität ausgerichtet ist. Dazu hat Abran als Software Measurement Framework den Metrology-Ansatz konzipiert $SMF_{Metrology}$ [Abran 2010]. Die vier Grundkomponenten dieses Ansatzes sind im Folgenden kurz charakterisiert.

Die Metrologie-Sichtweise: Es geht hierbei um die typischen Merkmale und Komponenten einer Messung überhaupt [VIM 2007]. So sollte beispielsweise auch ein Software-Mess-Tool die grundsätzlichen Eigenschaften eines Messgerätes schlechthin aufweisen (siehe folgende Abbildung).

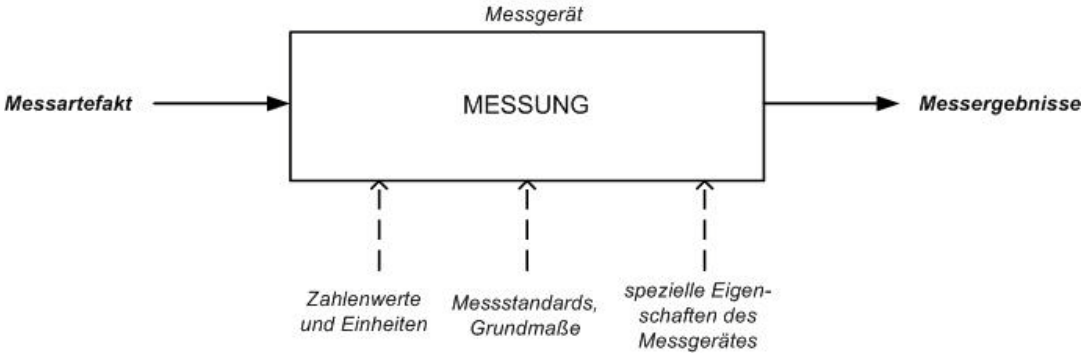


Abbildung 12: Messprozessmerkmale zu einem Messgerät

Der Software-Messkontext: Hiermit sind die typischen Merkmale eines Software-Messprozesses gemeint, wie wir sie bereits oben erläutert haben. Sie geben dem Metrology-Ansatz die softwaretechnische Ausrichtung.

Der praktische IT-Hintergrund: Hiermit sind Grundprinzipien und -inhalte gemeint, wie wir sie in den allgemeinen Management-Beschreibungen bzw. -Anleitungen [PMBOK 2004] und schließlich auch den praxisbezogenen Lehrinhalten ([Abran 2004], [Meridji 2012]) vorfinden.

Das Standard-Maß: Ähnlich einem Urmeter (Standard Etalon) sollte eine Grundmenge an Software-Maßen in einem metrologischen Kontext definiert und schließlich auch generell angewandt werden. Der Prozess dieser Standard-Maßbildung ist ausführlich in [Abran 2010] beschrieben.

Ein kurzes Fazit

Die Software-Messung und -Bewertung hat sich in den letzten 30 Jahren zu einem wichtigen und auch in seiner Ausprägung grundlegenden Disziplin des Software Engineering etabliert ([Balzert 2008], [Dumke 2003], [Ebert 2007], [Jones 2010], [Kandt 2006], [Kitchenham 2007], [Liggesmeyer 2002], [Sommerville 2010], [Zuse 1999]). Dazu können und wollen wir erst einmal kurz Folgendes konstatieren:

- Es existieren bereits viele Bausteine (Konzepte, Standards, Frameworks, Tools usw.) für den Einsatz von Messprozessen in der Praxis für die unterschiedlichsten Systemarten und Paradigmen. So ist beispielsweise die messbasierte Produktqualitätssicherung für klassische Paradigmen, wie der OOSE oder CBSE auch für den Wartungsbereich metriken- und tool-gestützt im Wesentlichen gesichert. Für aktuelle Paradigmen, wie den SOSE-Bereich oder gar AOSE-Bereich schon in wesentlichen Teilen gegeben.

- Hinsichtlich der wissenschaftlichen Darstellung haben wir vielfältige Diskussionen und Konzepte, die eine empirisch basierte Software-Bewertung im eigentlichen Sinne des Software Engineering untermauern.

- Es gibt eine motivierte Community zur Verbreitung von Erfahrungen in der Anwendung der Software-Messung und -Bewertung im praktischen Bereich als auch für die Identifikation neuer relevanter Forschungsaufgaben auf diesem Gebiet.

Bezüglich der Software Measurement Framework ergibt sich beispielsweise die folgende allgemeine Unterstützungsmöglichkeit für ein reales IT-Umfeld.

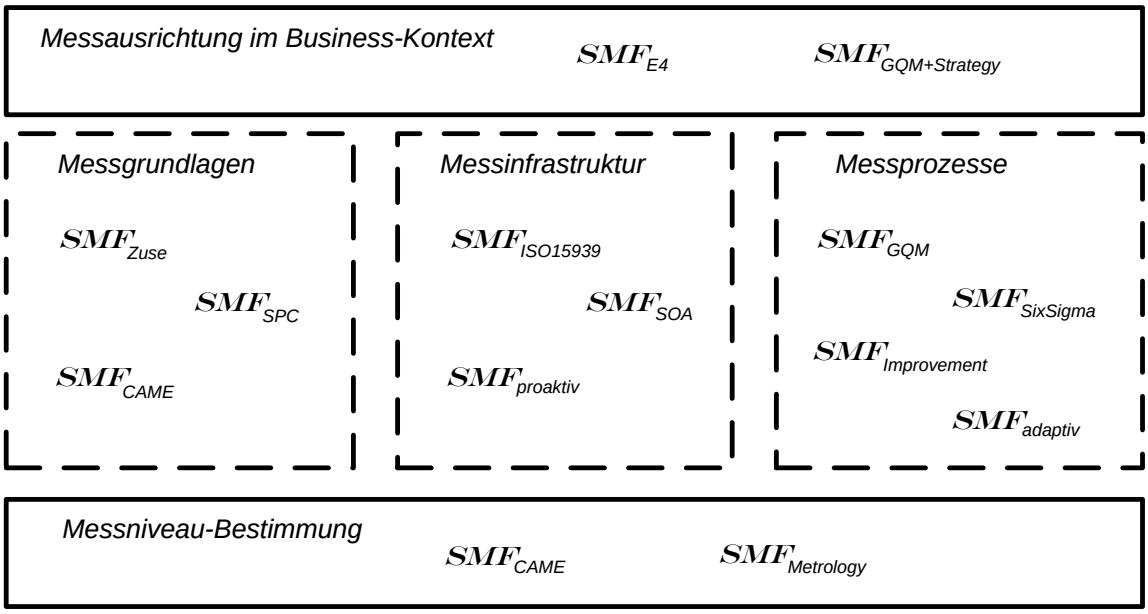


Abbildung 13: Unterstützungsbereiche der Software Measurement Frameworks

Diese geschlossen erscheinenden Konzepte und Methoden zur Software-Messung und -Bewertung täuschen allerdings nicht über einige, noch bestehende Probleme hinweg, die wir im Folgenden ganz allgemein charakterisieren wollen.

Offene Probleme der Software-Messung und -Bewertung

1. **Das Ignoranz-Problem:** Software und insbesondere Software-Prozesse werden noch zumeist nicht gemessen und sind damit auch nicht bewertbar. Das äußert sich häufig darin, dass die Postulierung neuer Methoden, Konzepte und Technologien *nicht quantifiziert untersetzt* ist. Das hat keinesfalls etwas mit Arroganz zu tun, sondern ist dem zumeist optimistischen Ausgangspunkt geschuldet, mit einem konstruktiven, allgemein konzipierten Ansatz ganze Problemklassen lösen zu können bzw. zu wollen. Das bedeutet beispielsweise, dass der Übergang von einem wohldefinierten Konstrukt bzw. Algorithmus zu einer nur noch beobachtbaren, auf Wahrscheinlichkeiten basierenden Implementation zu weit hinausgezögert oder gar ignoriert wird.

   Erste/weitere Schritte zur Lösung dieses Problems könnten sein:

- Von der Mess-Community sind weitere Aktivitäten für die IT-Probleme und deren Beherrschung zu informieren und zu motivieren notwendig.

- Es sind weitere Publikationsarbeiten auch in bzw. für die anderen Communities, die beispielsweise auf den Gebieten der *MDA* (Model Driven Architektur), der *STA* (Software Test und Analyse), der agilen Community u. a. m. vor allem mit der Anwendung der Software-Messung und -Bewertung zu tun haben, erforderlich. Dabei ergeben sich häufig auch neue, Community-spezifische Problemstellungen, die z. B. durch Messmethoden zu beantworten sind und somit wiederum die Mess-Community selbst bereichern.

- Es sollte stärker eine Projektrelevanz bei Forschungsvorhaben eingefordert werden. Das bedeutet beispielsweise, das man in zunehmenden Maße eine Validation der neuen Ansätze einfordern sollte, um auch einen wirklichen Wissensfortschritt nachzuweisen.

2. **Das Motivations-Problem:** Software-Messung ist zumeist freiwillig und wird noch zu häufig durch Pauschal-Annahmen und Leitlinien ersetzt, wie etwa "Qualität liegt in der Verantwortung jeden Mitarbeiters" oder auch "Die Wiederverwendung von Software-Komponenten sichert eine höhere Qualität" und andere haltlose bzw. in dieser allgemeinen Form nicht nachprüfbare Thesen. Man sollte sich ständig den möglichen Schaden durch Software vor Augen halten, wie zum Beispiel in der Risks-Sammlung von Neumann [Neumann 2012].

   Erste/weitere Schritte zur Lösung dieses Problems könnten sein:

   - Die zunehmende Dominanz der Software in allen Bereichen der Gesellschaft sollte vielmehr verdeutlicht werden (Verkehrsmitteln, Wahlen, Kriegsführung, Medizintechnik usw.), damit man sich gleichzeitig auch der Gefahren und der daraus resultierenden wachsenden Verantwortung bewusst ist.
   - Durch die bereits vorhandenen Erfahrungen bei der Software-Messung und -Bewertung sollten stärker vorhersagbare Probleme identifiziert und publiziert werden.
   - Die Software-Mess-Community sollte stärker Best Practices publizieren bzw. konsequenter das bereits vorhandene Wissen zur Software-Messung (Skalenklassifikationen, Messprozesse und -Frameworks) publizieren bzw. von neuen Technologien und Paradigmen einfordern.

3. **Das Qualitäts-Problem:** Die Software-Messung ist selbst noch zu wenig als allgemeingültige Form der Qualitätsbestimmung und -sicherung ausgeprägt. Das gilt einerseits für die Mess-Tools, die beispielsweise bei ein und derselben Metrik und demselben Artefakt unterschiedliche Messwerte liefern [Abran 2010]. Andererseits schlagen Standards - wie die ISO 9126:2003 - Metriken vor, denen man eine Verhältnisskala zuordnet, nur weil sich der Metrikenwert aus einem Zahlenverhältnis ergibt [ISO 2003]. Auszählungen werden dort generell als Absolutskala definiert. Und schließlich wird immer noch der Ansatz, alle Teilmessungen in einer einzigen Kennzahl zusammenfassen zu wollen, ohne korrekt die unterschiedlichen empirischen Verhältnisse zu beachten, präferiert.

   Erste/weitere Schritte zur Lösung dieses Problems könnten sein:

   - Es sind mehr, Tool-Herstellerunabhängige Analysen durchzuführen und zu publizieren, um die Sinnfälligkeit bzw. Notwendigkeit der Software-Messung und -Bewertung nicht zu untergraben bzw. in Frage zu stellen. Das gilt auch für neue Software-Entwicklungsansätze generell.
   - In der Mess-Community sind hier weitere Arbeitsgruppen-Aktivitäten (wie etwa die BSOA [Schmietendorf 2010]) oder auch Auftragsprojekte notwendig.
   - Man sollte mal wieder über Gütesiegel und ähnliche, institutionelle Bewertungsformen nachdenken.

4. **Das Kritikalitäts-Problem:** Die Bestimmung der Messziele selbst ist häufig noch zu wenig in seinem speziellen und allgemeinen Kontext definiert. Diese Problem wurde beispielsweise von Buglione als *Definitionsproblem* identifiziert [Buglione 2012] bzw. als Kernproblem empirischer

Bewertung in {Kitchenham 2007]. Häufig führt die Fragestellung des Messziels erst einmal zur Bestimmung der Merkmale und expliziten Beschreibung des Messartefaktes selbst. Die oben bereits genannte Eigenschaft der Software-Messung als "zweiten Schritt" verdeutlicht hierbei die unkorrekte bzw. unvollständige Ausführung des "ersten Schrittes" (Artefakterstellung und vor allem -dokumentation).

Erste/weitere Schritte zur Lösung dieses Problems könnten sein:

- Aufgrund der allgemeinen, globalen Bedeutung dieses Problems sollte beispielsweise die Mitarbeit in den entsprechenden (internationalen) Standardorganisationen kontinuierlich wahrgenommen werden, wie etwa zum *SMM* (Structured Metrics Meta-Model) der OMG (www.omg.org/spec/SMM/1.0/) oder bei der Cloud-Measurement-Initiative [CloudCommons 2010].
- Es sollten eigene Intiativen der Community, beispielsweise für "GQM Patterns" u. ä. m. unternommen werden.
- In diesem Zusammenhang sollten mehr Gütemodell- und Kriterienkataloganalysen zur Verbesserung der Transparenz auf diesem Gebiet durchgeführt werden.

5. **Das Komplexitäts-Problem:** Die zu messenden Artefakte (Produkt, Prozess, Ressourcen) sind bezüglich ihrer Komplexität noch zu wenig für die Messung beschrieben. Während man sich über die Problemstellung der algorithmischen Komplexität und den verschiedenen Ausprägungen (z. B. Roger's Gallery of Complexity) und zu lösenden Aufgaben ziemlich klar ist [Hemaspaandra 2002], ist die Software-Komplexitätsproblematik, wie beispielsweise die ca. 20 Programmkomplexitätsarten von Jones [Jones 2010] bzw. die Dokumentations- und Wartungskomplexitäten ([Lehner 1991], [Sneed 2010]) und deren Ausprägung für die verschiedenen Paradigmen noch unzureichend definiert und angewandt.

   Erste/weitere Schritte zur Lösung dieses Problems könnten sein:

   - Die oben charakterisierte Art und der Inhalt der Software-Komplexität, wie z. B. die Jones-Komplexitäten, sind auf die Prozesse zu erweitern, um die Komplexität der Artefakte besser in die Software-Messung und -Bewertung einzubeziehen.
   - Für die wirklichkeitsgetreue Berücksichtigung der Prozesskomplexität sollten mehr kausale Modelle definiert und validiert werden.
   - Hier sollte vor allem die "Theorie" (d. h. die Community der Theoretischen Informatik) motiviert werden, im obigen Sinne ihren möglichen und auf jeden Fall sinnvollen und wichtigen Beitrag zu leisten.

6. **Das Persistenz-Problem:** Software-Messmethoden sind noch zu wenig fester Bestandteil von Entwicklungs-, Wartungs-, Anwendungsmethoden und Tools. Das bedeutet einerseits, dass zum Teil CASE-Tools anstelle von Entwicklungsmetriken nur Code-Metriken zur Verfügung stellen und andererseits vor allem "Verlaufsmetriken" (z. B. Anforderungsverb -> Objektdienstmodell -> Dienstimplementation) zumeist gänzlich fehlen [Yazbek 2010].

   Erste/weitere Schritte zur Lösung dieses Problems könnten sein:

   - Hierbei erweist sich eine mögliche e-Measurement-Unterstützung durch Mess-Services als mögliche, flexible Lösung.
   - Außerdem sollte eine evolutionäre Messprozess- und Modellentwicklung iniiert werden, die die bedarfsgerechte Messintensität bzw. -ausprägung nach eigenen Bedarf selbst bestimmbar macht.
   - Schließlich sollte natürlich unbedingt mehr und besser bereits Erreichtes publiziert und kommuniziert werden.

7. **Das Grundmaßeinheiten-Problem:** Für eine Vergleichbarkeit über Produkte und Prozesse hinweg gibt es kaum Software-Metriken oder gar -Maße. In dieser Hinsicht kann man nur "neidisch" auf andere Wissenschaftsgebiete, wie die Mathematik oder Physik schauen, die hier schon eine Menge (allerdings über Jahrhunderte hinweg) erreicht haben. Ausgangspunkt ist natürlich die Notwendigkeit von Vergleichbarkeit und exakter Bewertung. Einer

ausdrucksvollen, mathematischen Darstellung, wie etwa $e^{i\pi} = -1$ [Bentley 2008], könnten wir bestenfalls ein

$$prozessmerkmale \times umfang^{produktmerkmale} = aufwand$$

oder gar die oben genannte Rayleigh-Kurve bezüglich einer Aufwandsverteilung entgegensetzen. Bei den Maßen allerdings, die wir zum Teil natürlich auch aus der Physik nutzen, sieht es da als "reine Software-Maße" noch schlechter aus.

Erste/weitere Schritte zur Lösung dieses Problems könnten sein:

- Bezüglich der Software-Maße sollte man offensiver bekannte, weit genutzte Maße (McCabe, FP usw.) kommunizieren und konseqent anwenden. Allerdings sollte hier auch der akuelle Kentnisstand zugrude gelegt werden. Die Untauglichkeit einiger C&K-Metriken bzw. einiger Halstead-Metriken ist lange bekannt und dennoch werden sie in Publikationen noch als Ausgangspunkt bzw. in Tools verwendet ([Ebert 2007], [Munson 2003], [Zuse 1998]).
- Bei der Charakterisierung von Software-Metriken und -Maßen sollten wohldefinierte Klassifikationen, wie z. B. die Skalenklassifikation von Zuse, strikter angewandt werden.
- Schließlich sollten weitere Metriken- und Maß-Initiativen insbesondere für aktuelle und moderne Paradigmen konsequent genutzt werden.

8. **Das Erfahrungs-Problem:** Die Schwellwerte für Metriken oder gar Maßeinheiten berücksichtigen zu wenig Paradigmen- oder Technologiebezogenheit. Außerdem sind ein Großteil der Erfahrungen firmenbezogene Fallstudien, die auch nur für dieses Umfeld weiter genutzt werden können ([Büren 2009], [Abran 2006], [Ebert 2004]). Verallgemeinerungen, wie zum Beispiel die Best Practices, haben dabei zumeist nur Orientierungscharakter ([Boehm 2007], [Johnson 2007], [Jones 2010], [Witt 2012]).

   Erste/weitere Schritte zur Lösung dieses Problems könnten sein:

   - Erfahrungsdatenbasen, wie u. B. die ISBSG, sollten erst einmal erschlossen und vor allem mit aktuelleren Projektdaten qualifiziert werden.
   - Außerdem sollten unbedingt Erfahrungen "vorangegangener" Paradigmen genutzt und für den neuen Anwendungsfall schnell angepasst werden.
   - Es sollten weitere Initiativen zu erfolgreichen Bewertungsformen intendiert, gefördert und publiziert werden.

9. **Das Kausalitäts-Problem:** Die Software-Messung berücksichtigt noch zu wenig sowohl methodische als auch technologische Prozessabläufe. Das schließt insbesondere den Entwicklungsverlauf, wie unter 6. bereits charakterisiert, mit ein und sollte vor allem die Zusammenhänge aller an der Entwicklung, Wartung und Anwendung beteiligten Artefakte in ihrer Kausalität zum Ergebnis so weit wie möglich einbeziehen.

   Erste/weitere Schritte zur Lösung dieses Problems könnten sein:

   - Hierzu sollte man zunächst auch die Tool-Hersteller mehr in die Pflicht nehmen. Pauschale Werbehinweise, wie etwa das Vorhandensein einer metrikenbasierten Ergebnisdarstellung sollten stets zum installierten Messprozess hinterfragt werden.
   - Dabei sind die Methoden- und Tool-Anforderungen expliziter zu beschreiben und in ihrer Integration deutlicher herauszustellen.
   - Letztlich kann die Software-Messung unter diesem Aspekt auch als ein Mittel zur "Prozessbereinigung" bzw. transparenteren Ausprägung dienen.

10. **Das Evidenz-Problem:** Software Engineering ist noch weit von einer axiomatischen Theorie realer Systeme entfernt. Vergleicht man die eleganten konsistenten und akkumulierten Darstellungen aus der Mathematik oder Physik, wie z. B. in den Büchern von Singh ([Singh 2005], [Singh 2005a]) kann man wiederum nur "neidisch" sein. Das drückt sich unter anderem in einer unzureichenden Evidenz vieler, insbesondere neuer Software-Engineering-Ansätze

aus, die bereits Definiertes oder Implementiertes ignorieren bzw. unzureichend in Vorhandenes einordnen (siehe [Erdogmus 2010], [Pfleeger 2005], [Zelkowitz 2007]). So vergleichen beispielsweise Publikationen zur agilen Software-Entwicklung ihren Ansatz häufig mit dem Wasserfall-Modell und verschweigen die Ähnlichkeit bzw. Gleichheit mit dem partizipatorischen, inkrementellen Ansatz, allerdings mit der agilen "Kernkonstanten", der Entwicklungszeitbeschränkung. Forschungen zu Produktlinien orientieren sich zum Teil an Programmversionen und nicht an Software-Produkten und können sich dadurch nicht als Weiterentwicklung des Konfigurationsmanagement verstehen bzw. auf diese wissenschaftlichen Ergebnisse aufbauen. Die strikte Trennung zwischen (funktionalem) Modell und der daraus konsistent abgeleiteten Architektur, wie sie bereits seit mehr als 20 Jahren im strukturierten SA/SD-Ansatz definiert und metrikenbasiert bewertbar ist, wird beim MDA-Ansatz nicht als Grundlage benutzt und weist diesen Ansatz nicht als spezifische Weiterentwicklung sondern zum Teil sogar als angeblich neu aus. Zwischen dem in der Theorie entwickelten Begriff der Programmkorrektheit mit den umfangreichen und leistungsfähigen Ansätzen zur Korrektheitsbeweis einer Programmbeschreibung und der tatsächlich implementierten Programmkorrektheit besteht immer noch keine begrifflich klare Trennung und daher beispielsweise ein Unverständnis für empirische Analysen und systematischem Vorgehen usw. usw. ([Boehm 2007], [Broy 2006], [Bundschuh 2008], [Dyba 2005], [Fitzgerald 2012], [Sommerville 2010]). So kann man natürlich bezüglich unserer Paradigmendarstellung in Abbildung 5 auch anderer Meinung sein, allerdings vorausgesetzt, man hat in all diesen Paradigmen bereits auf dem Level 4 gearbeitet und/oder valdiationsbasiert geforscht. So gilt für viele Bereiche die Feststellung von Zelkowitz "I find it very frustating when we are still talking about the same debugging techniques that were 'old' when I started teaching in 1971" [Zelkowitz 2012]. Auch in der letzte Computerzeitung auf Seite 8 steht: "Viele Trends leben vom Vergessen". Diese Schwäche des Software Engineering ist natürlich auch nachteilig für die Software-Messung und -Bewertung bezüglich Vergleichbarkeit von (historischen) Messdaten, die Anwendung von bereits quantifizierten Paradigmen-Prozessen bis hin zur (unnötig) beschränkten Gültigkeit der Messprozesse selbst.

Erste/weitere Schritte zur Lösung dieses Problems könnten sein:

- Hier ist unbedingt wieder die Theorie gefragt, sich stärker den komplexen Bereichen des Software Engineering auch in seiner ursprünglichen Bedeutung als Ingenieursdisziplin zu widmen, und zwar als ***Theorie realer Software-Systeme***. Dabei sollte ein axiomatischer Ansatz angestrebt werden, wie beispielsweise die Kombination von Architekturkonstruktion und Metriken als *Metrikenalgebra* von Shepperd oder Ejiogu ([Ejiogu 1991], [Shepperd 1995]) oder die axiomatische, graphbasierte Metrikendefinition von Fenton/Whitty und Prather ([Fenton 1997], [Prather 1984]).
- Initiativen in dieser Hinsicht sollten auch von der Mess-Community initiiert und vielleicht sogar moderiert werden. Institutionelle Voraussetzungen sind dafür gegeben ([Abran 2010], [Bouwers 2012], [Büren 2009], [Heidrich 2009], [Jones 2010], [Kunz 2010], [Sneed 2010]).
- Das gilt auch für die Ausbildung der Studenten bei der die Validation - als unbedingter Realitätsbezug - und die dafür vorhandenen Methoden und Ansätze der Software-Messung und -Bewertung unbedingt Lehrinhalt sein sollten ([Abran 2004], [Meridji 2012], [PMBOK 2004]).

Wie gesagt: Software-Messung und -Bewertung ist immer nur der zweite Schritt (oder auch nur Hilfsmittel) nach einer Software-Entwicklung und -Anwendung und kann und soll nur helfen, notwendige Entscheidungen und Maßnahmen möglichst korrekt bzw. qualifiziert zu treffen Dazu sind wirklich alle Informatiker aufgerufen, sich an der anstehenden bzw. notwendigen Erarbeitung von Wissen auf diesem Gebiet zu beteiligen. Unsere GI-Fachgruppe und der DASMA-Verein sind nur Katalysatoren für dieses Kerngebiet des Software Engineering.

Literatur

[Abran 2010] Abran, A.: *Software Metrics and Software Metrology. John Wiley & Sons* 2010

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

[Abran 2004] Abran, A.; Moore, J.: *Guide to the Software Engineering Body of Knowledge - SWEBOK*. Los Alamitos, California: IEEE Computer Society, 2004

[Awodey 2010] Awodey, S.: *Category Theory.* Oxford University Press. New York, 2010

[Bache 1994] Bache, R.; Bazzana, G.: *Software Metrics for Product Assessment.* McGraw Hill Publ., 1994

[Backhaus 2010] Backhaus et al.: *Multivariate Analysemethoden*. Springer Publ., 2010

[Balzert 2008] Balzert, H.: *Lehrbuch Softwaretechnik: Softwaremanagement.* Spektrum-Verlag, 2008

[Basili 1994] Basili, V; Caldiera, G., Rombach, D.: *The Goal Question Metric Approach.* Encyclopedia of Software Engneering, John Wiley & Sons, 1994, S. 528-532

[Basili 2007] Basili, V. R. et al.: *Empirical Software Engineering.* LNCS 4336, Springer Publ., 2007

[Basili 2007a] Basili, V. R. et al.: *GQM+Strategies.* In: Büren/Bundschuh/Dumke: Praxis der Software-Messung, Shaker Publ., Aachen, 2007, pp. 253-266

[Basili 1986] Basili, V. R.; Selby, R. W.; Hutchens, D. H.: *Experimentation in Software Engineering.* IEEE Transactions on Software Engineering, 12(1986)7, pp. 733-734

[Bentley 2008] Bentley, P. J.: *Das Buch der Zahlen.* Primus Verlag, Darmstadt, 2008

[Boehm 2000] Boehm, B. W.: *Software Cost Estimation with COCOMO II.* Prentice Hall Inc., 2000

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

[Boehm 1991] Boehm, B.: *Software Risk Management: Principles and Practices*. In IEEE Software, 1991

[Bouwers 2012] Bouwers, E.; Visser, J.; Deursen, Arie van: *Getting What You Measure.* Comm. of the ACM, 55(2012)7, pp. 54-59

[Braungarten 2005] Braungarten, R.; Kunz, M.; Dumke, R.: *An Approach to Classify Software Measurement Storage Facilities.* Preprint No 2, University of Magdeburg, Dept. of Computer Science, 2005

[Brooks 1975] Brooks, F. P.: *The Mythical Man Month - Essays on Software Engineering.* Addison-Wesley Publ., 1975

[Broy 2006] Broy, M.: *The 'Grand Challenge' in Informatics: Engineering Software-Intensive Systems.* IEEE Computer, October 2006, pp. 72-80

[Büren 2009] Büren, G. et al.: *Praxis der Software-Messung.* Shaker Verlag, Aachen, 2009

[Buglione 2012] Buglione, L.: *Measuring things right by the right things: balancing perspectives.* In: Schmietendorf/Wille: Beiträge zum empirischen Software Engineering - Eine Bilanz. Shaker-Verlag, 2012, S. 1-12

[Buglione 2003] Buglione, L.: *Misurare il software*. FracoAngeli Publ., Rom, 2003

[Bundschuh 2008] Bundschuh, M.; Dekkers, C.: *The IT Measurement Compendium.* Springer Publ., 2008

[Card 1990] Card, D.N.; Glass, R. L.: *Measuring Software Design Quality.* Prentice-Hall Inc., 1990

[Ciolkowski 2002] Ciolkowski, M.; Hartkopf, S.; Laitenberger, O.; Rombach, D.: *Das ViSEK-Projekt: Aufbau einer nationalen empirisch-basierten Erfahrungsdatenbank für Software-Engineering:* In: Dumke/Rombach: Software-Messung und -Bewertung, Deutscher Universitätsverlag, 2002, S. 1-12

[CloudCommons 2010] Cloud Commons: “The Details behind the Service Measurement Index“;<http://www.cloudcommons.com/servicemeasurementindex/-/asset_publisher/M69c/content/the-details-behind-the-service-measurement-index>, November 2010

[Damrauer 2008] Damrauer, C.: *Weltformeln.* Regensburg, 2008

[Davis 1995] Davis, A. M.: *201 Principles of Software Development.* McGraw Hill Publ., 1995

[DeMarco 1991] DeMarco. T.; Lister, T.: *Wien wartet auf Dich! Der Faktor Mensch im DV-Management.* Carl-Hanser Verlag, 1991

[Dyba 2005] Dyba, T.; Kitchenham, B.; Jorgensen, M.: *Evidence-Based Software Engineering for Practictioners.* IEEE Software, Jan./Feb. 2005, pp. 58-65

[Dumke 2011] Dumke, R. R.: *Product Portfolios Management.* In: P.A. Laplante: Encyclopedia of Software Engineering, Volume II, CRC Press Taylor & Francis Group, Boca Raton, USA, 2011, pp. 729-752

[Dumke 2009] Dumke, R.: *Service-orientierte Messdatenbanken.* In: Büren et al.: Praxis der Software-Messung, Shaker-Verlag, S. 17-28

[Dumke 2003] Dumke, R.: *Software Engineering.* 4. Auflage, Vieweg-Verlag, Braunschweig, 2003

[Dumke 1992] Dumke, R.: *Softwareentwicklung nach Maß - Schätzen, Messen, Bewerten.* Vieweg-Verlag Braunschweig, 1992

[Dumke 2011a] Dumke, R; Abran. A.: *COSMIC Function Points – Theory and Advanced Practices*. CRC Publ., Boca Raton, 2011

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

[Dumke 2008] Dumke, R. R. et al.: *Software Process und Product Measurement.* Springer Publ., 2008

[Dumke 1996] Dumke, R.; Foltin, E.; Koeppe, R.; Winkler, A: *Softwarequalität durch Meßtools - Assessment, Messung und instrumentiertes ISO 9000*. Vieweg-Verlag, Wiesbaden Braunschweig, Germany 1996

[Dumke 2008a] Dumke, R.; Kunz, M.; Farooq, A. Georgieva, K.; Hegewald, H.: *Formal Modelling of Software Measurement Levels of Paradigm-Based Approaches.* Technical Report FIN-013-2008, University of Magdeburg, 2008

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

[Dumke 2010] Dumke, R.; Mencke, S.; Wille, C.: *Quality Assurance of Agent-Based and Self-Managed Systems.* CRC Press, Boca Raton, 2010

[Dumke 2002] Dumke, R.; Rombach, D.: *Software-Messung und Bewertung.* Deutscher Universitätsverlag, Wiesbaden, 2002

[Dumke 1997] Dumke, R.; Winkler, A.: *Managing the Component-Based Software Engineering with Metrics*. Proc. of the Fifth International Symposium on Assessment of Software Tools, Pittsburgh, June 2-5, 1997, pp. 104-110

[Ebert 2012] Ebert, C.: *Project and Process Measurement. .* In. Schmietendorf/Wille: Beiträge zum empirischen Software Engineering - Eine Bilanz. Shaker-Verlag, 2012, S. 17-34

[Ebert 2011] Ebert C., Buglione L., *Estimation Tools - An overview*. In: Büren/Dumke/Münch: Praxis der Software-Messung. Shaker-Verlag, Aachen, 2011, S. 3-16

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

[Ebert 1996] Ebert, C.; Dumke, R.: *Software-Metriken in der Praxis.* Springer-Verlag, 1996

[Ebert 2004] Ebert, C.; Dumke, R.; Bundschuh, M.; Schmietendorf, A.: *Best Practices in Software Measurement.* Springer Publ., 2004

[Ejiogu 1991] Ejiogu, L.O.: *Software Engineering with Formal Metrics.* QED Technical Publ., 1991

[Emam 1998] Emam, K. E.; Drouin, J. N.; Melo, W.: *SPICE – The Theory and Practice of Software Process Improvement and Capability Determination.* IEEE Computer Society Press, 1998

[Endres 2003] Endres, A.; Rombach, D.: *A Handbook of Software and Systems Engineering*. Pearson Education Limited, 2003

[Erdogmus 2010] Erdogmus, H.: *How Important Is Evidence, Really?* IEEE Software, May/June 2010, pp. 2-4

[Ezran 1998] Ezran, M.; Morisio, M.; Tully, C.: *Pactical Software Reuse: the essential guide.* Freelife Publ., Paris, 1998

[Farooq 2005] Farooq, A*.: Conception and Prototypical Implementation of a Web Service as an Empirical-based Consulting about Java Technologies.* Master Thesis, University of Magdeburg, 2005 (im WEB unter http://poseidon.cs.uni-magdeburg.de:8080/oomj/ verfügbar)

[Fehlmann 2010] Fehlmann, T.: *From Story Points to COSMIC Function Points in Agile Development - A Six Sigma Perspective.* In: Abran et al.: Applied Software Measurement, Shaker-Verlag, 2010, S. 349-360

[Fenton 1997] Fenton, N. E.; Pfleeger, S. L.: *Software Metrics - a rigorous and practical approach.* Thompson Publ., 1997

[Fitzgerald 2012] Fitzgerald, B.: *Software Crisis 2.0.* IEEE Computer, April 2012, pp. 89-91

[Florac 1999] Florac, W. A.; Carleton, A. D.: *Measuring the Software Process – Statistical Process Control for Software Process Improvement.* Pearson Education, 1999

[Frohnhoff 2009] Frohnhoff, S.: *Use Case Point 3.0 - Implementierung einer Use Case bezogenen Schätzmethode für das Software-Engineering betrieblicher Informationssysteme.* Universität Paderborn, 2009

[Garcia 2011] Garcia, F.; Vizcaino, A.; Ebert, C.: *Process Management Tools.* IEEE Software, March/April 2011, pp. 15-18

[Georgieva 2010] Georgieva, K.: *Testing of Aspect Oriented Programs - Testing Aspect Oriented Programs as Object Oriented Ones.* Lambert Academic Publishing AG & Co. KG Saarbrücken, 2010

[Georgieva 2009] Georgieva, K.; Dumke, R. Neumann, R.; Farooq, A.: *Software Measurement Modelling and Improvement.* Proc. of the SERP 2009, Las Vegas, July 2009, pp. 396-402

[Gilb 1977] Gilb, T.: *Software Metrics.* Winthrop Publ., 1977

[Glaeser 2010] Glaeser, G.; Polthier, K.: *Bilder der Mathematik.* Spektrum-Verlag, 2010

[Goodman 1993] Goodman, P.: *The Practical Implementation of Software Metrics*. McGraw-Hill Inc., 1993

[Halstead 1977] Halstead, M. H.: *Elements of Software Science.* Prentice Hall, New York, 1977

[Heidrich 2009] Heidrich, J.; Münch, J.; Wickenkamp, J*.: Practical Guidelines for Introducing Software Cockpits in Industry.* Proceedings of the 5th Software Measurement European Forum. 2009.S. 49-64

[Hemaspaandra 2002] Hemaspaandra, L. A.; Ogihara, M.: *The Complexity Theory Companion.* Springer Publ., 2002

[Henderson-Sellers 1996] Henderson-Sellers, B.: *Object-Oriented Metrics: Measures of Complexity.* Prentice-Hall Inc., 1996

[Hollocker 1990] Hollocker, C. P.: *Software Reviews and Audits Handbook.* John Wiley & Sons Inc., 1990

[Huberman 2001] Huberman, B. A.: *The Laws of the Web – Pattern in the Ecology of Information.* MIT Press, Cambridge, Mass., 2001

[Humphrey 2000] Humphrey, W. S.: *The Personal Software Process: Status and Trends*. IEEE Software, Nov/Dec. 2000, pp. 71-75

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

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

[ISO 2003] ISO/IEC 9126: *Software Engineering – Product Quality*. 2003

[Isozimov 1989] Isozimov, A.: *Metricheskaja Ozenka Katschestva Programm.* Verlag Isdatelstvo Mai, Moskau, 1989

[Johnson 2007] Johnson, B.; Higgins, J.: *ITIL and the Software Lifecycle: Pactical Strategy and Design Principles.* Van Haren Publ., Amerfoort, Netherland, 2007

[Jones 1991] Jones, C.: *Applied Software Measurement.* McGraw-Hill, 1991

[Jones 2010] Jones, C.: *Software Engineering Best Practices – Lessons from Successful Projects in the Top Companies.* McGraw-Hill Publ., 2010

[Jones 1986] Jones, C. B.: *Systematic Software Development Using VDM.* Prentice Hall Publ., 1986

[Juristo 2003] Juristo, N.; Moreno, A. M.: *Basics of Software Engineering Experimentation*. Kluwer Academic Publishers, Boston, 2003

[Kan 1995] Kan, S. H.: *Metrics and Models in Software Quality Engineering.* Addison-Wesley Publ., 1995

[Kandt 2006] Kandt, R. K.: *Software Engineering Quality Practices.* Auerbach Publications, Boca Raton New York, 2006

[Kenett 1999] Kenett, R. S.; Baker, E. R.: *Software Process Quality – Management and Control.* Marcel Dekker Inc., 1999

[Keyes 2003] Keyes, J.: *Software Engineering Handbook*. Auerbach Publ., 2003

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

[Kitchenham 1996] Kitchenham, B. A.: *Software Metrics – Measurement for Software Process Improvement.* NCC Blackwell Ltd, 1996

[Kulpa 2003] Kulpa, M. K.; Johnson, K. A.: *Interpreting the CMMI – A Process Improvement Approach.* CRC Press Company, 2003

[Kunz 2010] Kunz, M.: *Framework for a Service-Oriented Measurement Infrastructure.* Shaker Publ., Aachen, Germany 2010

[Laird 2006] Laird, L. M; Brennan, M. C.: *Software Measurement and Estimation: A Practical Approach.* John Wiley & Sons Publ., 2006

[Lanza 2006] Lanza, M.; Marinescu, R.: *Object-Oriented Metrics in Practices.* Springer Publ., 2006

[Lehner 1991] Lehner, F.: *Softwarewartung - Management, Organisation und methodische Unterstützung*. Carl-Hanser-Verlag, 1991

[Lewerentz 2002] Lewerentz, C.: Metrics-Based Quality Analysis of Large Software Products in Dumke/Bundschuh: Software-Metriken in der Praxis, Shaker Verlag, Aachen, 2002, S. 133-146

[Liggesmeyer 2002] Liggesmeyer, P-: *Software-Qualität: Testen, Analysieren und Verifizieren von Software.* Spektrum Verlag, 2002

[Liu 2008] Liu, P.; Du, Z.: *E-commerce Performance Assessment Research Based on the D-S Theory and the Balanced Score Card Method*. Proc. of the 2008 International Seminar on Future Information Technology and Management Engineering, IEEE Computer Press, pp. 120-123

[Lorenz 1994] Lorenz, M.; Kidd, J.: *Object-Oriented Software Metrics.* Prentice Hall Publ., 1994

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

[Masak 2009] Masak, D.: *Digitale Ökosysteme*. Springer Publ. 2009

[McCabe 1976] McCabe, T.: *A Complexity Measure.* IEEE Transactions on Software Engineering, 2(1976)4, S. 308-320

[McConnel 2006] McConnel, S.: *Software Estimation – Demystifying the Black Art.* Microsoft Press, 2006

[McGarry 2002] McGarry, J. et al.: *Practical Software Measurement – Objective Information for Decision Making.* Addison Wesley Publ., 2002

[Menasce 2002] Menasce, D. A.: *QoS Issues in Web Services*. IEEE Internet Computing, Dec. 2002, pp. 72-75

[Menasce 1998] Menasce, D. A.; Almeida, V. A. F.: *Capacity Planning For Web Performance - Metrics, Models, & Methods.* Prentice Hall, 1998

[Mencke 2008] Mencke, S.: *Proactive Ontology-Based Content Provision in the Context of e-Learning.* PhD thesis, University of Magdeburg, Dept. of Computer Science, Germany, 2008

[Meridji 2012] Meridji, K. et al.: *A Framework for the Measurement Concepts in the SWEBOK Guide.* In: Schmietendorf/Wille: Beiträge zum empirischen Software Engineering - Eine Bilanz. Shaker-Verlag, 2012, S. 85-92

[Munson 2003] Munson, J., C.: *Software Engineering Measurement.* CRC Press Company, Boca Raton London New York, 2003

[Neumann 2012] Neumann, P. G.: *Computer Related Risks.* Addison Wesley Publ., 1995, aktuell unter www.csl.sri.com/~neumann/ (letzter Zugriff; 5. August 2012)

[Neumann 2010] Neumann, R.: *Orthogonale Softwaremaße in der Fehlerprognose.* Verlag Dr. Hut, München, 2010

[Neumann 2012a] Neumann, R.: *The Internet of Products: An Approach to Establishing Total Transparency in Electronic Markets.* Springer-Verlag, 2012 (erscheint demnächst)

[Neumann 2009] Neumann, R.; Zbrog, F.; Dumke, R.: *Cockpit-Based Enterprise Application Development.* In: Büren et al.: Praxis der Software-Messung, Shaker Publ., 2009, pp. 235-246

[Nielson 1999] Nielson, F.; Nielson, H. R.; Hankin, C.: *Principles of Program Analysis.* Springer Publ., 1999

[Ormandjieva 2010] Ormandjieva, O.; Mikhnovsky, V.; Yurasovskaya, E.; Klasa, S.: Bentahar, .: *Categorical Representation of Software Services Quality and Decision Making Using Markov Decision Process.* In: Abran et al.: Applied Software Measurement, Shaker Publ., Aachen, Germany, 2010, pp. 35-46

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

[Pearl 2000] Pearl, J.: *Causality – Models, Reasoning, and Inference.* Cambridge University Press, 2000

[Peled 2001] Peled, D. A.: *Software Reliability Methods.* Springer Publ., 2001

[Pfleeger 1998] Pfleeger, S. L.: *Software Engineering - Theory and Practice.* Prentice-Hall Publ., 1998

[Pfleeger 2005] Pfleeger, S. L.: *The Role of Evidential Force in Empirical Software Engineering.* IEEE Software, Jan./Febr. 2005, pp. 66-73

[PMBOK 2004] PMBOK: *A Guide to the Project Management Body of Knowledge,* Third Edition. Project Management Institute, 2004

[Potter 2000] Potter, R. W.: *The Art of Measurement.* Prentice Hall Publ., 2000

[Poulin 2001] Poulin, J.S.: *Measurement and Metrics for Software Components*, in Heineman, G. T.; Councill, W. T. (editors): Component-based Software Engineering: putting the pieces together, Addison-Wesley, Boston, 2001

[Prather 1984] Prather, R.E.: *An Axiomatic Theory of Software Complexity Measure.* The Computer Journal, 27(1984)4, pp. 340-347

[Prechelt 2001] Prechelt, L.: *Kontrollierte Experimente in der Softwaretechnik - Potenzial und Methode.* Springer-Verlag 2001

[Putnam 2003] Putnam, L. H.; Myers, W.: *Five Core Metrics – The Intelligence Behind Successful Software Management.* Dorset House Publishing, New York, 2003

[Rajlich 2006] Rajlich, V.: *Changing the Paradigm of Software Engineering.* CACM 49(2006)8, pp. 67-70

[Richter 2012] Richter, K.: *Modeling, Evaluating and Predicting of IT Human Resources Performance.* Dissertationsschrift, Universität Magdeburg, Fakultät für Informatik, 2012

[Robillard 1985] Robillard, P.:*Le Logiciel: de sa conception a sa maintenance*. Gaetan Morin Verlag, Quebec, 1985

[Royce 1998] Royce, W.: *Software Project Management.* Addison-Wesley, 1998

[Rud 2009] Rud, D.: *Performancebewertung und -sicherung von orchestrierten Serviceangeboten.* Verlag Dr. Kovac, Hamburg, 2009

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

[Schmietendorf 2012] Schmietendorf, A.: *Bedeutung der Softwaremessung bei agilen Projektparadigmen - Eine Bestandsaufnahme.* In: Schmietendorf/Wille: Beiträge zum empirischen Software Engineering - Eine Bilanz. Shaker-Verlag, 2012, S. 93-106

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

[Schmietendorf 2001] Schmietendorf, A.: *Process Concepts Ensuring Software Performance Engineering in Large IT-Organizations (German)*. Shaker Publ., Aachen, Germany, 2001

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

[Schmietendorf 2010] Schmietendorf, A. et al.: *BSOA 2010 – Bewertungsaspekte serviceorientierter Architekturen.* Shaker Publ., Aachen, Germany 2010

[Shepperd 1995] Shepperd, M.: *Foundations of Software Measurement.* Prentice Hall Publ., 1995

[Shneiderman 1980] Shneiderman, B.: *Software Psychology.* Winthrop Publ., Mass. 1980

[Simon 2001] Simon, F.: *Messwertbasierte Qualitätssicherung - Ein generisches Distanzmaß zur Erweiterung bisheriger Softwareproduktmaße.* Dissertationsschrift, TU Cottbus, 2001

[Singh 2005] Singh, M.: *Big Bang - Der Ursprung des Kosmos und die Erfindung der modernen Naturwissenschaft.* Carl Hanser Verlag München, 2005

[Singh 2005a] Singh, M.: *Fermats Letzter Satz - Die abenteuerlichen Geschichte eines mathematischen Rätsels..* dtv Verlag , 2005

[Singpurwalla 1999] Singpurwalla, N. D.; Wilson, S. P.: *Statistical Methods in Software Engineering.* Springer Publ., 1999

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

[SML@b 2012] SML@b-Portal: www.smlab.de

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

[Sneed 2010] Sneed, H.; Seidl, R.; Baumgartner, M.: *Software in Zahlen - Die Vermessung von Applikationen.* Carl-Hanser Verlag, 2010

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

[Sommerville 2010] Sommerville, I.: *Software Engineering.* Pearson Publ., 2010

[SPC 1995] Software Productivity Consortium: *The Software Measurement Guidebook.* Thomson Computer Press, 1995

[Symons 1991] Symons, C.: *Software Sizing and Estimating; MK II FPA.* John Wiley & Sons, 1991

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

[Tetlow 2007 Tetlow, P.: *The Web's Awake – An Introduction to the Field of Web Science and the Concept of Web Life*. IEEE Press, New York, 2007

[Thaller 1994] Thaller, G. E.: *Software-Metriken - einsetzen, bewerten, messen.* Heise-Verlag, Hannover 1994

[VIM 2007] VIM ISO/IEC Guide 99: International vocabulary of metrology – Basic and general concepts and associated terms (VIM). International Organization for Standardization – ISO, Geneve 2007

[Vollmann 1990] Vollmann, S.: *Aufwandschätzung im Software Engineering.* IWT Verlag, München, 1990

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

[Wang 2000] Wang, Y.; King, G.: *Software Engineering Processes – Principles and Applications.* CRC Press, 2000

[Welke 2011] Welke, R.; Hirschheim, R.; Schwar, A.: *Service-Oriented Architecture Maturity.* IEEE Computer, February 2011, pp. 61-67

{Whitmire 1997] Whitmire, S.A.: *Object Oriented Design Measurement.* John Wiley & Sons, 1997

[Whitty 1990] Whitty, R. W.: *Structural Metrics for Z Specifications.* Proc. of the Z User Workshop, Oxford 1989, Springer Publ., 1990, pp. 186-191

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

[Wille 2010] Wlle, C.; Kuhl, .; Dumke, R.; Kun, M.: *Evidence-Based Classification and Evaluation of Effort Estimation Methods (German).* Abran et al.: Applied Software Measurement, Shaker Publ., Aachen, Germany, 2010, pp. 169-186

[Witt 2012] Witt, T. C.: *IT Best Practices - Management, Teams, Quality, Performance, and Projects.* CRC Press, Boca Raton, 2012

[Wohlin 2000] Wohlin, C, Runeson, P, Höst, M, Ohlsson, M, Regnell, B, Wesslén, A.: *Experimentation in Software Engineering: An Introduction*. Kluwer Academic Publishers, Boston, 2000

[Yazbek 2010] Yazbek, H.: *Framework of Quality Measurement in CASE Tool Based Software Development.* Dissertationsschrift, Universität Magdeburg, Fakultät für Informatik, 2010

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

[Zelkowitz 2012] Zelkowitz, M. V.: *What have we Learned About Software Engineering.* Comm. of the ACM, 55(2012)2, pp. 38-39

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

[Zuse 1991] Zuse, H.: *Software Complexity – Measures and Methods*. De Gruyter Publ., Berlin, 1991

[Zuse 1999] Zuse, H.: *Thirty Years of Software Measurement.* In: Dumke/Abran: Software Measurement – Current Trends in Research and Practice, DUV, 1999, pp. 3-37

Witt, T.C.:

IT Best Practices: Management, Teams, Quality, Performance, and Projects

CRC Press, 2012 (355 Seiten), ISBN: 978-1-4398-6854-6

Ideal for project, IT, and systems development managers, IT Best Practices: Management, Teams, Quality, Performance, and Projects covers the skills, knowledge, and attributes needed to succeed in bringing about large-scale change. It explains how to incorporate quality methods into the change management process and outlines a holistic approach for transformation management.
Detailing time-tested project management techniques, the book examines management skills with a focus on systems thinking to offer a pragmatic look at effecting change. Its comprehensive coverage spans team building, quality, project methodology, resource allocation, process engineering, and management best practices. The material covered is validated with references to concepts and processes from such business greats as W. Edwards Deming, Jack Welch, and Henry Ford. Readers will learn the history behind the concepts discussed along with the contributions made by these great business minds.
The text supplies an awareness of the factors that impact performance in today's projects to supply you with the real-world insight needed to bring about large-scale change in your organization. Although it is geared around change, most of the concepts discussed can be directly applied to improve efficiencies in your day-to-day activities.

Büren, G.; Dumke, R.R.; Münch, J.:

MetriKon 2011 - Praxis der Softwaremessung
Tagungsband des DASMA Software Metrik Kongresses
17.-18. November 2011, Kaiserslautern

Shaker Verlag, Aachen, 2011 (210 Seiten), ISBN 978-3-8440-0557-8, ISSN 1618-7946

The book includes the proceedings of the MetriKon 2011 held in Kaiserslautern in November 2011, which constitute a collection of theoretical studies in the field of software measurement and case reports on the application of software metrics in companies and universities.

Schmietendorf, A.; Simon, F.:

BSOA 2011
6. Workshop Bewertungsaspekte serviceorientierter Architekturen
15. November 2011, Köln

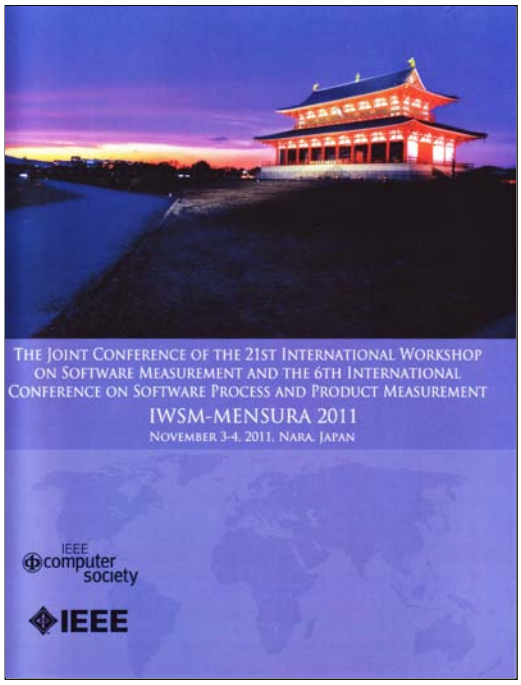
Shaker Verlag, Aachen, 2011 (108 Seiten), ISBN 978-3-8440-0503-5, ISSN 1867-7088

Seit nunmehr 6 Jahren beschäftigt sich die BSOA-Initiative mit der Bewertung von serviceorientierten Architekturansätzen. Zunächst beschäftigten sich die Teilnehmer im Rahmen der ersten Workshops mit der messtechnischen Erfassung der mit einer SOA einhergehenden Ausprägungen und Merkmale bzw. den involvierten Stakeholdern. Sehr schnell wurde deutlich, dass sich eine SOA weniger auf technologische Sachverhalte bezieht als vielmehr auf die veränderte Sichtweise zur Gestaltung unternehmensweit genutzter IT-Systeme. Erwartete Vorteile einer SOA bezogen sich insbesondere auf die Zielstellungen des Informationsmanagements. In diesem Zusammenhang wurden Mehrwertpotentiale durch eine verbesserte Geschäftsprozessorientierung der IT, reduzierte Daten- und Funktionsredundanzen, verringerte Komplexitäten bei Anwendungen und Schnittstellen, verringerte Kundenbindungen oder auch die Flexibilität mit der eine benötigte IT-Lösung bereitgestellt werden kann, ausgemacht.
Aus der Vielzahl an eingereichten Beiträgen konnte durch das Programmkomitee eine anspruchsvolle Agenda zusammengestellt werden.

IWSM-MENSURA 2011
The Joint Conference of the 21st International Workshop
on Software Measurement (IWSM) and the 6th International Conference
on Software Process and Product Measurement (MENSURA)
November 3-4, 2011, Nara, Japan

IEEE Computer Society Los Alamitos, California, Washington, Tokyo, 2011 (324 Seiten)
ISBN 978-0-7695-4497-7

This proceedings includes the full papers and the short papers of the Joint Conference of the 21st International Workshop on Software Measurement (IWSM) and the 6th International Conference on Software Process and Product Measurement (MENSURA).

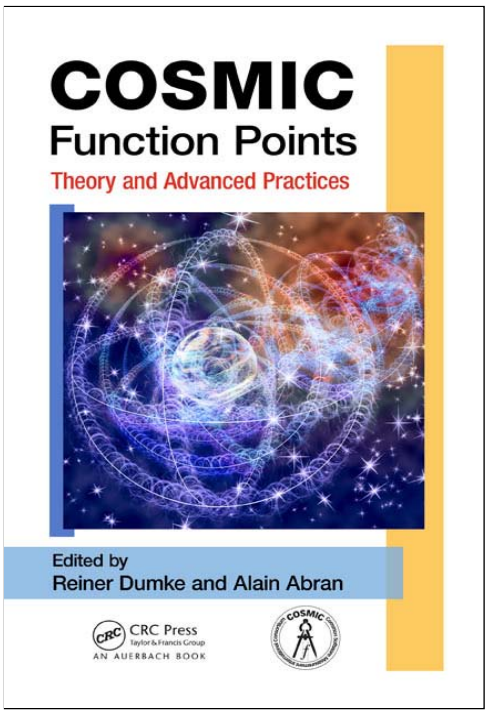


Dumke, R.; Abran, A.:

Cosmic Function Points
Theory and Advanced Practices

CRC Press Taylor & Francis Group, 2011 (334 Seiten), ISBN: 978-1-4398-4486-1

This book has the following characteristics: the theme is about a new software size estimation method including their scientific and practical background; the chapters are based on papers, that would be published in our conference proceedings during the last six years; the authors are wellknown participants of the international software measurement community (see e. g. COSMIC, IFPUG etc.) and the book content is structured in the main problems of building new measurement or estimation methods in general and should be interesting for the software quality.

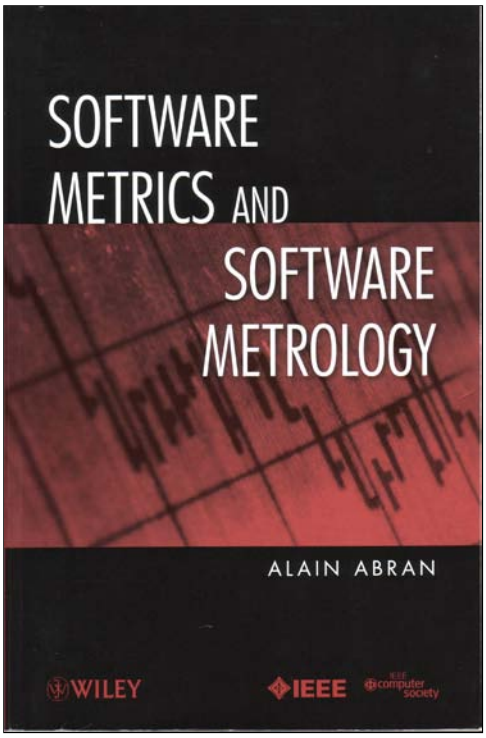


Abran, A.:

Software Metrics and Software Metrology

John Wiley & Sons, 2010 (328 Seiten), ISBN: 978-0-470-59720-0

Hundreds of software measures have been proposed to the industry over the past forty years, but limitations in their design mean that most of them are of insufficient use to practitioners. So how can practitioners recognize which measures are sound and useful for decision-making? Moreover, for those who have to design new software measures, what must they do to build sound ones?
Software Metrics and Software Metrology looks at the fundamentals of the design of a measurement method, which forms the foundation of the measures available in the sciences and in engineering. Alain Abran provides a step-by-step approach to both analyzing the design of current software measures and designing new, robust software measures for a specific business or engineering need. He draws upon years of experience to ensure that software engineers and managers will apply the best practices in software measurement - and therefore be equipped to respond to the most demanding customers and feel supported by senior executives.
Presents the key concepts that dictate whether a software measure's design is sufficiently strong. Features several case studies analyzing strengths and weaknesses in the design of some of the software measures most widely used or quoted. Describes how lessons learned led to the design of the COSMIC - ISO 19761 method for the measurement of the functional size of software, from its initial prototype to its adoption by the ISO as an international standard. Illustrates day-to-day software measurement issues that have not been seriously addressed, from convertibility across measurement designs to measurement standard etalons. Includes chapter exercises for classroom us.
Software Metrics and Software Metrology is meant for software quality specialists and process improvement analysts and managers, in software organizations of all sizes. In addition, this book introduces many of the theoretical concepts and references needed by professionals, managers, and students to help them understand the fundamentals of the identification and evaluation of software development and maintenance processes, as well as improvements to them.



ASQT 2012:

Arbeitskonferenz Softwarequalität und Test
September 6.-7., 2012, Klagenfurt, Austria
see: **http://www.asqt.org/**

CONQUEST 2012:

13. International Conference on Software Quality
September , 2012, Nuremberg, Germany
see: **https://www.isqi.org/de/isqi-news/items/conquest-in-nuernberg.html**

ESEM 2012:

International Symposium on Empirical Software Engineering & Measurement
September 20-21, 2012, Lund, Sweden
see: **http://www.esem-conferences.org/**

UKSMA 2012:

Annual UKSMA Conference - Managing your Software (through Measurement)
October , 2012, London, UK
see: **http://www.uksma.co.uk/**

IWSM/Mensura 2012:

Common International Conference on Software Measurement
October 17-19, 2012, Assisi, Italy
see: **http://iwsm2012.wordpress.com/**

BSOA 2012:

7. Workshop Bewertungsaspekte service-orientierte Architekturen
November 15, 2012, Desden, Germany
see: **http://www-ivs.cs.uni-magdeburg.de/~gi-bsoa/**

MetriKon 2012:

International Conference on Software Measurement
November 7 - 9, 2012, Stuttgart, Germany
see: **http://www.metrikon.de/**

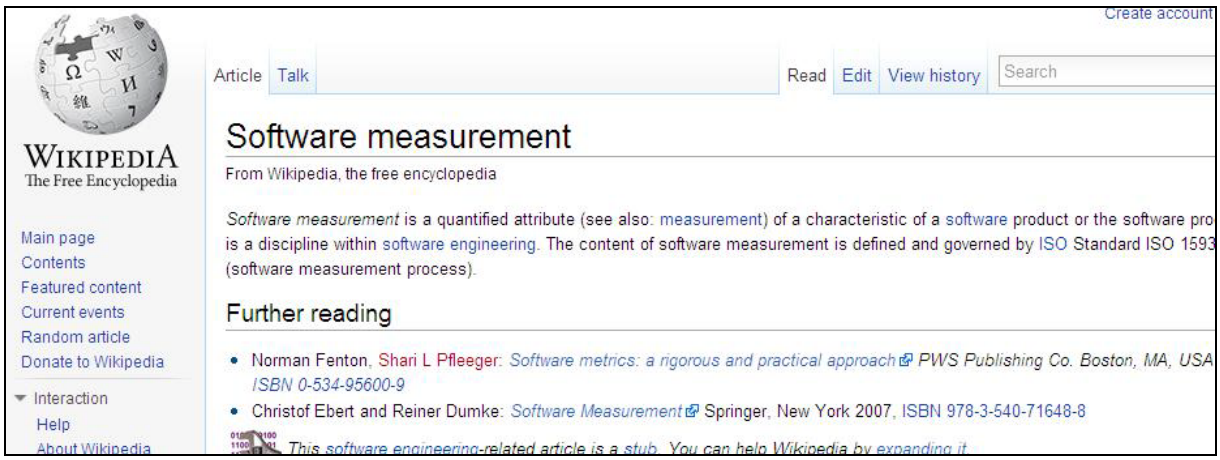
ICPE 2013:

International Conference on Performance Engineering
Spring 2013
see: **http://icpe.ipd.kit.edu:80/call_for_proposals/**

see also: OOIS, ECOOP and ESEC European Conferences

Hinweis: unsere neue GI-Webseite ist direkt zu erreichen unter **http://fg-metriken.gi.de/**

Außerdem ist unsere Thematik im Wikipedia noch zu wenig präsentiert. Während wir eine erste Charakterisierung zum Software Measurement haben (s. u.) fehlt zur Software-Messung und -Bewertung noch jegliche Beschreibung im Wikipedia.



Hier sollte unsere Community noch stärker wirksam werden.

SOFTWARE MEASUREMENT NEWS

VOLUME 17 2012 NUMBER 2

CONTENTS

Announcements **3**

Position Papers **11**

Abran, A.; Symons, C.:
COSMIC Annual Report 2011 - A year of great progress..... 11

Schmietendorf, A.; Neumann, R.:
Improvement the persistence layer inside well established software applications..... 18

Dumke, R. R.:
Software-Messung und -Bewertung - Eine Bilanz..... 24

New Books on Software Metrics **57**

Conferences Addressing Metrics Issues **61**

Metrics in the World-Wide Web **63**

ISSN 1867-9196