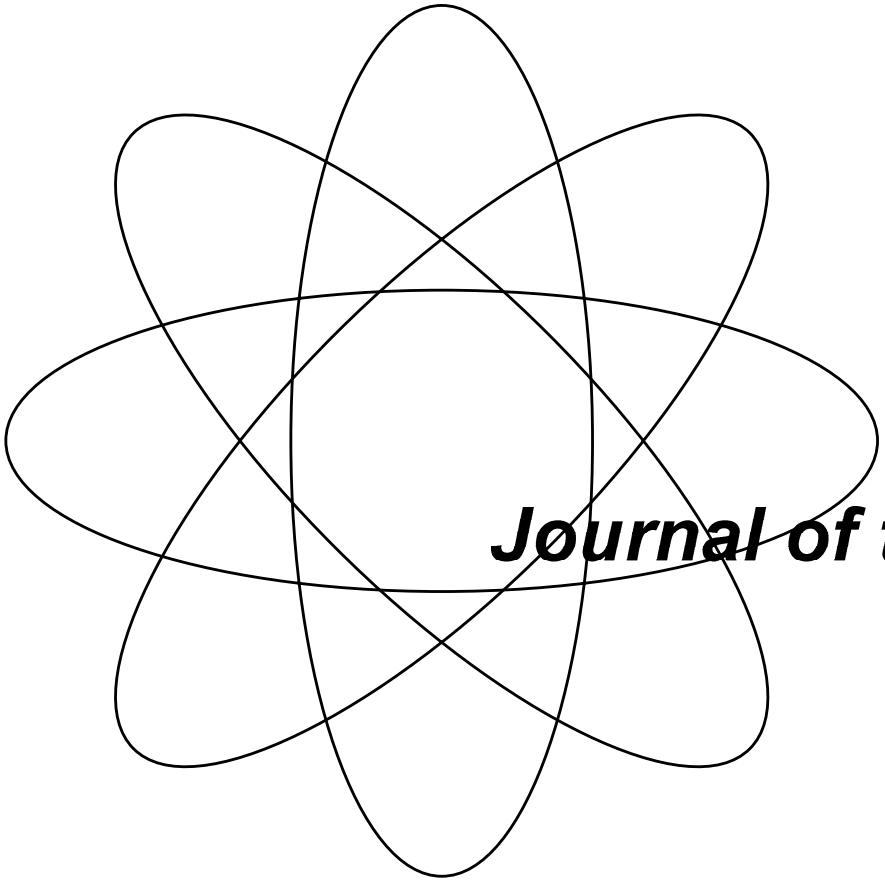
METRICS NEWS

Volume 12, Nur



Journal of the Softw

Mea

Analyzing

The *METRICS 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

Manfred Bundschuh
Chair of the DASMA
Sander Höhe 5, 51465 Bergisch Gladbach, Germany
***Tel.:** +49-2202-35719*
manfred.bundschuh@netcologne.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

Horst Zuse
Dr.-Ing. habil. in Computer Science
Technical University of Berlin, FR 5-3,
Franklinstr. 28/29, D-10587 Berlin, Germany
***Tel.:** +49-30-314-73439, **Fax:** +49-30-314-21103*
zuse@tubvm.cs.tu-berlin.de

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

Technical Editor: DI Martin Kunz

The journal is published in one volume per year consisting of two numbers.

==========

C A L L F O R P A P E R S

==========

17th International Workshop on Software Measurement
IWSM 2007

& MENSURA 2007

Co-sponsored by:

École de Technologie Supérieure - Université du Québec
(Montréal, Canada)
Otto von Guericke University Magdeburg (Germany)
Universidad de los Baleares (Spain)
Universidad de Alcala de Henares (Spain)
Universidad del Pais Vasco (Spain)

In cooperation with:

COSMIC – Common Software Measurement International Consortium
German Interest Group on Software Metrics

Nov. 5-7, 2007

Palma de Majorque - SPAIN

==========

GENERAL THEME & SCOPE: SOFTWARE MEASUREMENT

Software measurement is one of the key technologies to control or to manage the software development process. Measurement is also the foundation of both sciences and engineering, and much more research in software is needed to ensure that software engineering be recognized as a true engineering discipline.

Over the past few years, a significant number of key institutional documents have been brought into the public domain with a broad consensus based as ISO standards and technical reports. Therefore, it is necessary to exchange between researchers and practitioners the experiences on the design and uses of measurement methods to simulate further theoretical investigations to improve the engineering foundations through measurement.

The purpose of the conference is to review the set of issues such as the identification of deficiencies in the design of currently available measurement methods, the identification of design criteria and techniques and measurement frameworks.

We are looking for full papers in the area of software measurement, addressing generic research issues, infrastructure issues or specific research and implementation issues on the following topics (but not limited to):

A- Uses of measurements results in decision making:

- Productivity Analysis (foundations of productivity models, quality of productivity models, experimental basis and constraints that limit it expandability to contexts outside of the experimental basis).
- Estimation process (uncertainty, identification of inputs, expectations, technical estimates versus business risks estimation, etc.).

B- Evaluation and assessment models:

- Performance assessment
- Quality assessment
- Maintenance assessment
- *Support systems assessment*

C- Objects and attributes to be measured

- Types of measurement object targets: functional domains, type of software – layers, specific functional characteristics – algorithms.
- Timely adaptation of the designs of measurement methods to new and emerging technologies: UML, Web-based applications, Agent based systems, etc.
- Size attributes categories: Functional and non-functional, etc.

D- Measurement methods: design issues

- Design issues of measurement methods: definition of base components to be measured, ISO conformance, weights assignments and theoretical foundations (Basis for consensus, degree of consensus, etc.
- Normalization issues: time dependence, technology dependence, infrastructure changes
- Integration of measurement types: when and how.
- Quality of measurement methods: repeatability accuracy, correctness, traceability, uncertainty, precision, etc.

Special Activities on Monday Nov. 5, 2007

To participate to the workshops, the participants must submit a Position Paper by the specified deadline.

Workshop 1:

- Software Measurement Body of Knowledge (Basis for discussion: the Draft of a Measurement Body of Knowledge on the IEEE SWEBOK website)

Workshop 2:

- Fundamentals principles of software engineering

Certification exam:
COSMIC-FFP – ISO 19761: Entry Level

PROGRAM COMMITTEE

Alain Abran, École de technologie supérieure - Université du Québec, Canada
Ali Idri, INSIAS, Morocco,
Luigi Buglione, AtosOrigin, Italy
Manfred Bundschuh, DASMA, Germany
François Coallier, ÉTS, Canada
Juan Cuadrado Gallego, U. Alcala de Henares, Spain
Jean-Marc Desharnais, ÉTS, Canada
Javier Dolado, Universidad San Sebastian, Spain
Ton Dekkers, Shell, Netherlands
Reiner Dumke, University of Magdeburg, Germany
Christof Ebert, Vector Consulting, Stuttgart, Germany
Naji Habra, FUNDP, Namur, Belgium
Nadine, Hanebutte, St. John Fisher College, Rochester, USA
Adel Khelifi, U. Al Hosn, URA
Mathias Lother, Bosch, Germany
Roberto Meli, DPO, Italy
Olga Ormandjieva, Concordia University, Canada
Mercedez Ruiz Carrera, U. Cadiz, Spain,
Andreas Schmietendorf, FHW Berlin, Germany
Harry Sneed, Anecon Wien/Budapest, Hungary
Charles Symons, Software Measurement Service Ltd, Edenbridge, UK
Hannu Toivonen, Nokia, Finland
Horst Zuse, TU Berlin, Germany

SUBMISSIONS – <u>Research Track</u> – Full papers

Authors should send proposed papers by e-mail to the conference co-chairs

- Deadline for proposed papers: **May 30, 2007**
- Notification of acceptance: **June 20, 2007**
- Paper - final version for the proceedings: **Sept 15, 2007**

SUBMISSIONS – <u>Industry Track</u> (presentations only)

- Abstract (max. 1 page): **Sept 15, 2007**
- Notification of acceptance on: **Sept 30, 2007**
- Final Powerpoint presentation: **Oct 15, 2007**

Position papers (1 to 5 pages) for the workshops on Nov. 5

- Deadline: **Sept 15, 2007**

All proposals should be sent to:

Alain Abran **alain.abran@etsmtl.ca** École de technologie Supérieure		*Reiner Dumke* **dumke@ivs.cs.uni-magdeburg.de** Otto-von-Guericke-Universitaet Magdeburg

FEES for authors: to be determined
FEES for participants: to be determined

NEWS: For the latest news about IWSM-MENSURA 2007, see:

http://gelog.etsmtl.ca/iwsm-mensura2007

Deutschsprachige
Anwendergruppe für
Software-
Metrik und
Aufwandschätzung e.V.

MetriKon 2007

DASMA Software Metrik Kongress

15. / 16. November 2007

Fraunhofer IESE Kaiserslautern

GI-Fachgruppe 2.1.10
"Software Messung
und Bewertung"

CALL FOR PAPERS

Veranstalter

DASMA e.V. und GI-Fachgruppe 2.1.10

Kontaktadressen für

MetriKon-Beiträge:
metrikon-beitraege@dasma.org

Ausstellungsreservierungen:
DASMA e.V.
c/o Romy Robra
Haidbuckel 17
90542 Eckental
Fon: 09126 / 29 79 576
Fax: 09126 / 28 24 43
E-Mail: info@dasma.org

Programmkomitee

Manfred Bundschuh
DASMA

Günter Büren
Büren & Partner, Nürnberg

Dr. Axel Dold
DaimlerChrysler AG, Ulm

Prof. Dr. Reiner Dumke
Universität Magdeburg

Dr. Christof Ebert
Vector Consulting, Stuttgart

Bernd Gebhard
Bayerische Motoren Werke, München

Prof. Dr. Hans-Georg Hopf
GSO-Fachhochschule Nürnberg

Dr.-Ing. Marek Leszak
Alcatel-Lucent, Nürnberg

Prof. Dr. Claus Lewerentz
Technische Universität Cottbus

Prof. Dr. Peter Liggesmeyer
Fraunhofer IESE, Kaiserslautern

Dr.-Ing. Mathias Lother
Robert Bosch GmbH, Stuttgart

Dr. Dirk Meyerhoff
Schüco-Service GmbH, Bielefeld

Dr. Jürgen Münch
Fraunhofer IESE, Kaiserslautern

Dr. Frances Paulisch
Siemens AG, München

Prof. Dr. Andreas Schmietendorf
Hochschule für Wirtschaft, Berlin

Harry Sneed
SES, München/Budapest

Dr. Cornelius Wille
FH Bingen

Prof. Dr.-Ing. Horst Zuse
Technische Universität Berlin

Themenstellung & Abgrenzung

Software-Messverfahren und Metriken sind Schlüsseltechnologien für Controlling und Management von Software-Entwicklungsprozessen und -Produkten. Messen ist eine wichtige Grundlage für jede Ingenieurtätigkeit und die Gewinnung neuer Erkenntnisse in Wissenschaft und Technik. Es ist damit sowohl für die Praxis der Software-Entwicklung als auch für die empirische Forschung zur Software-Technik unverzichtbar. Die MetriKon bietet für den notwendigen Erfahrungsaustausch zwischen Theorie und Praxis eine ideale Plattform.

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

MetriKon-Beiträge

Gewünscht werden **Erfahrungsberichte aus der industriellen Praxis** zur Einführung und Anwendung von Messprogrammen und zur Umsetzung von Messmodellen, sowie **Forschungsbeiträge** zur fachlichen Fundierung, Anwendung und Validierung von praxisrelevanten Softwaremetriken.

Ebenso können am Vortag der Tagung (14. November 2007) **Tutorien** zur Einführung oder Vertiefung komplexer Themen oder Verfahren der Softwaremessung und Aufwandschätzung angeboten werden. Vorschläge mit einer Kurzbeschreibung bitte bis zum 9. Mai 2007 einreichen.

Begleitend zur Tagung wird eine **Ausstellung** organisiert, in der Dienstleister und Werkzeughersteller den Teilnehmern ihre Angebote zur Unterstützung von Softwaremessung und Aufwandschätzung vorstellen. Aussteller sollten Ihren Teilnahmewunsch spätestens bis Ende August beim Veranstalter anmelden.

Themenfelder

Generell sind Beiträge zu allen Themen rund um Software-Metriken und Aufwandschätzverfahren erwünscht. Das Programmkomitee behält sich bei der Auswahl der eingereichten Beiträge vor, auf eine ausgewogene Mischung von Praxis und Theorie zu achten.

Zur Orientierung hier einige Stichworte zu erwünschten Vortragsthemen:

- Software-Metriken, Vergleich von Metriken, Einführung von Metriken etc.
- Zielorientierte Metriken zur Verfolgung und Erreichung von Projekt- und Verbesserungszielen
- Einsatz, Einführung und Erfahrungen mit Aufwandschätzverfahren und -Tools
- Quantitatives Projektmanagement, Projektcontrolling
- CMMI- / SPICE-konforme Messprogramme und Metriken
- Verwandte Themen wie Benchmarking von Prozessen und Projekten
- Einsatz von Mess- und Aufwandschätzverfahren im Zusammenhang mit Embedded Systems, SOA, Testen, Anforderungsmanagement, etc.

Weitere Informationen

Zur DASMA e.V. und zur MetriKon 2007 -Tagung finden Sie weitere Informationen im Web unter **http://dasma.org** und **http://www.metrikon.de**, zur GI-Fachgruppe 2.1.10 unter **http://ivs.cs.uni-magdeburg.de/sw-eng/us/giak**.

Beachten Sie bei der Ausarbeitung von Beiträgen für die MetriKon 2007 bitte die „Richtlinien zur Einreichung von Beiträgen zur MetriKon 2007". Sie finden diese Richtlinien und auch den Call for Papers im Internet auf den Seiten der MetriKon unter der oben angegebenen Adresse.

Wichtige Termine

18. Juni 2007	Abgabeschluss einer aussagekräftigen Kurzfassung
bis 31. Juli 2007	Benachrichtigung über die Annahme
17. September 2007	Abgabe des druckfertigen Beitrages

Call for Papers
Evaluation of Service-Oriented Architectures
BSOA 2007

Aufruf zur Einreichung von Beiträgen zum 2. Workshop
„*Bewertungsaspekte serviceorientierter Architekturen*“
der GI FG „Software-Messung und -Bewertung“
im November 2007

MOTIVATION

Glaubt man den Aussagen führender IT-Analysen, wie z.B. der Gartner-Group, so wird die Etablierung serviceorientierter Architekturen (kurz SOA) die Vorgehensweise bei der Entwicklung neuer Anwendungssysteme in den kommenden Jahren grundlegend beeinflussen. In diesem Kontext wird nicht selten vom Ende der Dominanz monolithischer Softwarearchitekturen gesprochen. Hintergrund dieser Überlegung ist die Tatsache, dass bei einer servicebasierten Architektur neue Anforderungen primär auf der Basis bereits existierender Serviceangebote realisiert werden können. Vor diesem Hintergrund werden neue Bewertungsmodelle benötigt, die prozess-, produkt- und ressourcenbezogene Aspekte im Kontext einer SOA berücksichtigen. Der Workshop (BSOA07) wird sich unter anderem mit den folgenden Themen beschäftigen:

- Bewertung der Mehrwertpotenziale einer SOA
- Erarbeitung von Richtlinien zu Serviceentwicklung für eine SOA
- Qualitätsbewertung angebotener Services
- Mess- und Bewertungsansätze im Kontext einer SOA
- Services Level Agreements (SLAs) und Verhandlungsaspekte

WORKSHOP-BEITRÄGE

Praktiker und Wissenschaftler, die auf dem Gebiet der Konzeption, Entwicklung und Management serviceorientierter Architekturen tätig sind, werden gebeten, Beiträge im doc- oder pdf-Format einzureichen. Der Umfang der Beiträge sollte 3000 Wörter nicht übersteigen. Die Formatierungsrichtlinien werden in Kürze auf der unten genannten Webseite veröffentlicht. Angenommene Beiträge werden innerhalb eines 30-minütigen Vortrags präsentiert bzw. in Form eines Posters vorgestellt. Angenommene Beiträge erscheinen in einem Tagungsband.

Bitte senden Sie ihre Beiträge per E-Mail an

`gi-bsoa@ivs.cs.uni-magdeburg.de`

WEBSEITE ZUM WORKSHOP

`http://ivs.cs.uni-magdeburg.de/~gi-bsoa`

Our 16th Workshop on Software Measurement (IWSM 2006) and DASMA Software Metrik Kongress (MetriKon 2006) took place in Potsdam, Germany in November 2006. The following report gives an overview about the presented papers. Furthermore, the papers are published in the following Shaker book (ISBN 3-8322-5611-3):

Magdeburger Schriften zum Empirischen Software Engineering

Eds.: Prof. Dr. Alain Abran, Université du Québec
Manfred Bundschuh, Vorsitzender der DASMA e.V.
Günter Büren, Büren & Partner Software-Design, Nürnberg
Prof. Dr.-Ing. habil. Reiner R. Dumke, Universität Magdeburg

OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG
Fakultät für Informatik
Institut für Verteilte Systeme
Arbeitsgruppe Softwaretechnik

Applied Software Measurement

Proceedings of the International Workshop on Software Metrics and DASMA Software Metrik Kongress

IWSM/MetriKon 2006

2.-3. November 2006,
Hasso-Plattner-Institut Potsdam, Germany

DASMA Deutschsprachige Anwendergruppe für Software-Metrik und Aufwandsschätzung

GI-Fachgruppe 2.1.10
Software Messung und Bewertung

Otto-von-Guericke-Universität Magdeburg
Software Measurement Laboratory (SMLab)

SHAKER VERLAG

KEYNOTE:

Establishing a Common Measurement System at Siemens

Frances Paulisch

Siemens AG Corporate Technology SE

Abstract. Software is of large and growing importance for practically all Siemens Groups. With ca. 30,000 software engineers worldwide it is clean that Software is an integral part of many of our products. Having adequate processes and the importance of process improvement activities has been an important topic at Siemens over the past decade and many Siemens organizations are firm believers in using measurement System to help control and improve the processes.
More recently, in the fall of 2004, we started the development of a Siemens-wide measurement System and this System is meanwhile also in broad use at Siemens. Many different sources of Information were taken into account, both Siemens-internal as well as external, to establish an approach that best meets the needs of a broad set of stakeholders.
The harmonized and common system within Siemens enables more transparency and allows increased best-practice sharing across organizations and Groups. This presentation will provide lessons learned in establishing such a measurement System and will describe the structure of the measurement system. Furthermore, first insights of what we can learn from the data will be given.

Enhancing the CoBRA® Hybrid Software Cost Modeling Method for Supporting Process Maturation

Adam Trendowicz, Jens Heidrich, Jürgen Münch

Fraunhofer Institute for Experimental Software Engineering (IESE)

{adam.trendowicz, jens.heidrich, juergen.muench}@iese.fraunhofer.de

Abstract. Cost estimation is a very crucial field for software developing companies. In the context of learning organizations, estimation applicability and accuracy are not the only acceptance criteria. The contribution of an estimation technique to the understanding and maturing of related organizational processes (such as identification of cost and productivity factors, measurement, data validation, model validation, model maintenance) has recently been gaining increasing importance. Yet, most of the proposed cost modeling approaches provide software engineers with hardly any assistance in supporting related processes. Insufficient support is provided for validating created cost models (including underlying data collection processes) or, if valid models are obtained, for applying them to achieve an organization's objectives such as improved productivity or reduced schedule. This paper presents an enhancement of the CoBRA® cost modeling method by systematically including additional quantitative methods into iterative analysisfeedback cycles. Applied at Oki Electric Industry Co., Ltd., Japan, the CoBRA® method contributed to the achievement of the following objectives, including: (1) maturation of existing measurement processes, (2) increased expertise of Oki software project decision makers regarding cost-related software processes, and, finally, (3) reduction of initial estimation error from an initial 120% down to 14%.

Using Genetic Algorithms to Generate Estimation Models

D. Rodríguez[1], *J.J. Cuadrado-Gallego*[1], *J. Aguilar*[2]

The University of Alcalá[1] / University Pablo de Olavide[2]

drg@ieee.org, jjcg@uah.es, direscinf@upo.es

Abstract. Parametric software estimation models rely on the availability of historical project databases from which estimation models are derived. In the case of large project databases, problems can arise such as heteroscedasticity where the size of a project can influence the accuracy of the estimation method. In such cases, a single mathematical model may not properly be used to estimate projects of diverse nature. In this work, we discuss how genetic algorithms can be applied to produce segmented models, i.e., the genetic algorithm searches for cut-points in the range of a variable (e.g. Function Points), and different estimation models can be used at each side of the cut-point. A concrete case study using the ISBSG dataset is reported. Results show that with a very low number of models instead of a single one, the accuracy can be increased significantly.

An Experimental Study on Conceptual Data Model Based Software Code Size Estimation

Oguz Atak[1], *Cigdem Gencel*[2]

[1]Havelsan Inc.,

oatak@sbd.havelsan.com.tr

[2]Informatics Institute-METU

cgencel@ii.metu.edu.tr

Abstract. Effort and cost estimation is crucial in software management. Estimation of software size plays a key role in the estimation process. SLOC has been a commonly used software size metric. However, SLOC of a software project is available only after the coding phase. We need to estimate SLOC early in the life cycle in order to make reliable effort and cost estimation, which are crucial at the beginning of a project.

Being an early phase product of the life cycle and being widely used during the requirements elicitation process of OO systems, the use of Conceptual Data Model for estimating SLOC have been explored in a number of studies. In this study, we explore whether the Conceptual Data Model can serve as an early indicator of software size by conducting an empirical study on two sample projects, which have similar characteristics and developed by the same software company.

Traceability zwischen Metriken und dem strategischen Ziel Wartbarkeit

Dr. Frank Simon, Christian Koll

SQS Software Quality System AG, Köln

(Frank.Simon | Christian.Koll)@sqs.de

Zusammenfassung: In diesem Beitrag wird aufgezeigt, wie mittels spezieller Modellierungstechniken strategische IT-Ziele wie die Wartbarkeit und konkrete, verfügbare Software-Metriken derart in Relation zueinander gebracht werden können, daß eine nachvollziehbare Traceability zwischen diesen beiden Stoßrichtungen hergestellt werden kann: Das Management erhält so auf Metriken basierende Aussagen (in Form von Aggregationen) bzgl. der Erfüllung ihrer IT-Strategien. Gleichzeitig besitzt die operative Ebene jederzeit die konkreten konstruktiven Stellschrauben, die bei Abweichungen vom Soll bedient werden müssen und anschließend wieder entsprechend in der Aggregation Berücksichtigung finden. Die berichteten Erfahrungen zeigen die Potentiale dieses Vorgehens auf.

Software Quality Assessment – A Tool-Supported Model

Matthias Ruffer[1], *Marek Leszak*[2]

[1]Friedrich-Alexander University Erlangen-Nürnberg

matthias_ruffler@fastmail.fm

[2]Lucent Technologies Network Systems GmbH, Nürnberg

mleszak@lucent.com

Abstract. This paper reports results of a practical diploma thesis, focusing on improvements to the currently introduced software quality assessment methodology (SQAM) at a large telecommunications supplier company. For this purpose the concept of assessing quality related process steps by quality gate reviews is extended by a flexible weighting scheme: Each software sub-team delivery can be evaluated quantitatively w.r.t. the process compliance reached. A model to calculate a so-called process compliance index (PCI) is introduced, based on the quality-related activities monitored. It is intended as an instrument to predict the quality level reached in future project releases. This requires the redesign of an existing toolset to support PCI calculation. The toolset is also being extended regarding aspects like central data management and modular expandability. To the authors' best knowledge there is no publication and no software engineering standard, dealing in-depth with this topic.

A Lightweight Tool Support for Integrated Software Measurement

Bernhard Daubner[1], *Andreas Henrich*[2], *Bernhard Westfechtel*[1]

[1]Bayreuth University, Chair of Software Engineering

bernhard.daubner@uni-bayreuth.de
bernhard.westfechtel@uni-bayreuth.de

[2]Bamberg University, Chair of Media Informatics

andreas.henrich@wiai.uni-bamberg.de

Abstract. This article shows a lightweight approach to implement a tool supported software measurement process. The basic idea here is to concentrate on five relevant software measures and tie them on the elements of an implicitly given skeletal structure of the project.We show several ways to provide such a skeletal structure for various types of software projects.
Based on this skeletal structure the software measures to collect can be defined in advance and independently of a concrete project. At project runtime the entities to measure are automatically identified by means of the elements of the skeletal structure. The computation of the software measures can then be automated using Open Source tools.

The call profile – measuring the object-oriented paradigm at work

Peter Rosner

London South Bank University, London

rosnerpe@lsbu.ac.uk

Abstract. In this paper we describe the call profile, a metric that gives an insight into different aspects of the object oriented paradigm at work in a piece of objectoriented software. It enables an estimation of the skill levels needed for those involved in its maintenance and evolution. A tool to measure the call profile for Java systems is described and some initial results are presented.

Ontology-based Web service for object-oriented metrics

Martin Kunz , Steffen Kernchen, Reiner R. Dumke, Andreas Schmietendorf

University of Magdeburg, Germany

{makunz, kernchen, dumke, schmiete}@ivs.cs.uni-magdeburg.de

Abstract. *The increasing economic relevance of software measurement for organizations* cannot be negated. But issues like complexity and missing traceability of measurement processes constitute the need for direction and guidance in this regard.
In history ontologies possessed the capability to retain this semantic knowledge in a machine-accessible manner. Therefore, we use the ontology approach for a cataloguing web system to create our own ontology for a subset of metrics (object oriented metrics).
In our approach the ontology is used to connect an information need with a certain metric.
We describe the interrelation of key elements like information need, measurement model, software characteristics, and object oriented structure.

Complexity and Quality Evaluation of Basic Java Technologies

Ayaz Farooq[1], Steffen Kernchen[1], Martin Kunz[1], Reiner R. Dumke[1], Cornelius Wille[2]

[1]University of Magdeburg, Faculty of Computer Science, Magdeburg, Germany

(farooq, kernchen, makunz, dumke)@ivs.cs.uni-magdeburg.de

[2]University of Applied Sciences, Bingen, Germany

wille@fh-bingen.de

Abstract. As a fact, the application of object-oriented approach is of high significance in the area of software development since it can abet efficiency or cost effectiveness and reduce error probability during software design and implementation. In order to quantify, especially qualitative aspects such as potential error hot spots caused by elevated design complexity, software measurement can strongly assist. Particularly, metrics proposed by Chidamber and Kemerer as well as Abreu's MOOD metrics set are presumably most prevalent in practice and provide adequate explanatory power. Especially the object-oriented programming language Java cannot be dismissed from one's thoughts because a lot of Java libraries serve as foundation for contemporary applications. Since Java technologies are widely used in industrial system applications, development of the complexity of different Java technologies could be an essential aspect in order to maintain the plenty set of existing Java applications successful. Therefore, we have analyzed several standard Java technology libraries in order to investigate such important characteristics. We have applied our approach using OOMJ web service as a case study to evaluate and analyze Java software products and standard libraries thereby highlighting their various complexity and quality aspects.

Object Relational Database Metrics: Classified and Evaluated

Justus S[1], Iyakutti K[2]

[1]Dept of Computer Applications, K.L.N College of Information Technology,
Pottapalayam – 630611, Sivagangai Dt, TN, India.

juskutti@yahoo.com, justus_mku@yahoo.co.in

[2]Dept of Microprocessor and Computer, School of Physics,
Madurai Kamaraj University, Madurai – 625021, TN, India.

Abstract. In the modern Object Oriented Information Systems, databases also have become a crucial object of concern. Object relational databases, often addressed as the next generation of databases, are complex in their existence because they combine relational database characteristics with object-oriented principles. Proposed object relational database metrics have been validated and is stated that it contributes to the quality design of the database and better functioning of the software.
This work presents a classified framework of object relation database metrics which deals in detail the semantics of the classes, relations, behavior with the application and its reusability. Classification of metrics institutes their better management of the database at their design phase, implementation phase and its behavior in the real time system developmental environment. The classification is validated for its dependability based on the evaluated experimental values. This classification is best admitted that database designers find better understanding of the Object relational database in entirety.

Unified Software Method: Towards a Method of Measurement of the Necessary Changes to Software in Maintenance

Stéphane Mercier, Alain Abran, Michel Lavoie, Roger Champagne

École de technologie supérieure, Montréal Québec, Canada

stephane.mercier.5@ens.etsmtl.ca, alain.abran@etsmtl.ca, michel.lavoie@etsmtl.ca, roger.champagne@etsmtl.ca

Abstract. Within the context of the use of the “Unified Software Method” (USM), traceability links are identified between each data element of a software project having a relation between them. One is then in the presence of a complete traceability which implies maintains it synchronization of information in a software project. In this article we propose a method of measurement based on USM and which aims at quantifying the quantity of information of a software project which is related to a maintenance action envisaged on an existing element in this project. One will be able to note that this method of measurement makes it possible to quantify at the same time the proportion of the information of the project as well as the quantity of information implied in maintenance considered.

Assessment Results using the Software Maintenance Maturity Model (S3m)

David-Alexandre Paquette, Alain April, Alain Abran

École de Technologie Supérieure

david-alexandre.paquette.1@ens.etsmtl.ca alain.april@etsmtl.ca, alain.abran@etsmtl.ca

Abstract. This **S**^{3m} maintenance maturity assessment model is divided into four process domains containing 18 "Key Process Area", each in turn containing "Roadmaps". Roadmaps are bodies of knowledge containing recommended practices that are linked to one another. Using the S^{3m} software maintenance maturity model, this paper describes the assessment process and results of an individual maintainer process maintaining a key software application within a larger software maintenance organization.

Product Metrics for Service-Oriented Infrastructures

Dmytro Rud, Andreas Schmietendorf, Reiner R. Dumke

Otto von Guericke University, Magdeburg, Germany

{rud, schmiete, dumke}@ivs.cs.uni-magdeburg.de

Abstract. Service-oriented architecture is nowadays widely adopted as modern approach for development of enterprise-wide and cross-enterprise distributed applications. From the software engineering point of view, these applications resemble some features of formerly known component-based and object-oriented software systems and web applications, but the differences are substantial enough to make it impossible to simply reuse existing metrics. In this paper we will try to analyse these differences and to formulate product metrics that consider all peculiarities of service-oriented software and assess its complexity, reliability and performance aspects.

Evaluation of Java-Based Agent Technologies

Steffen Kernchen[1], Ayaz Farooq[1], Reiner R. Dumke[1], Cornelius Wille[2]

[1]University Magdeburg, Faculty of Computer Science, Germany

{kernchen, farooq, dumke}@ivs.cs.uni-magdeburg.de

[2]University of Applied Sciences, Bingen, Germany

wille@fh-bingen.de

Abstract. Currently, the object-oriented approach (OOSE) is well-known and well-used in many industrial applications. Today, most of the problems with object-orientation are understood and some of the illusions of the "OO hype" are going in more realistic OO methods and OO techniques. In the same manner we can observe today the future technology of agent-oriented software engineering (AOSE).
Agent-oriented technology has been revisited as a complementary approach to the object-oriented paradigm, and has been applied in a wide range of realistic application domains, including e-commerce, human-computer interfaces, telecommunications, and concurrent engineering.
Our paper gives an analysis of the AOSE considering three types of AOSE technologies including their platforms: Aglets, MadKit and the JADE system. We use a Java Measurement Service which allows us to execute some complexity metrics like size metrics, Chidamber & Kemerer metrics and Abreu's MOOD.

Analyse struktureller Komplexitätsunterschiede in ABAP und JAVA

Roland Neumann, Alexandra Ilina

Technische Universität Kaiserslautern, AG Softwareengineering: Dependability

roland.neumann@informatik.uni-kl.de, alex.ilina@gmx.net

Zusammenfassung: Um große Softwaresysteme geeignet analysieren und warten zu können wird der Einsatz von Softwaremaßen immer wichtiger. Die Korrelation der Maße untereinander behindert dabei eine geeignete Auswertung. Mithilfe voneinander unabhängiger Maße läßt sich strukturelle Komplexität besser erfassen und einfacher auswerten.
Im Rahmen dieser Studie werden aus mehreren JAVA- und ABAP-Projekten sprachspezifische Komplexitätsarten identifiziert, ihre Gemeinsamkeiten und Unterschiede beschrieben und Nutzungsmöglichkeiten aufgezeigt.Gemeinsam in beiden Sprachen sind Größe und Attributverwendung, während sie sich in Kontrollfluß und Interaktion unterscheiden. Die vorgestellte Technik ermöglicht eine einfache Analyse von Klasseneigenschaften zur Identifikation diskreter Gruppen, was eine schnelle Inspektion und Fehleranalyse ermöglicht.

Suggestions for Improving Measurement Plans: A BMP application in Turkey

Luigi Buglione[1], Cigdem Gencel[2], Pinar Efe[3]

[1]École de Technologie Supérieure (ETS) Montréal, Canada

Luigi.buglione@computer.org

[2]Informatics Institute, Middle East Technical University - Ankara, Turkey

cgencel@ii.metu.edu.tr

[3]Siemens PSE Turkey - ODTU Teknokent Silikon Bina ZK 21 Ankara, Turkey

pinar.efe@siemens.com

Abstract. Time and Cost are most often in industry the two main (often solely) dimensions of analysis against which a project is monitored and controlled, excluding other possible dimensions such as quality, risks, impact on society and stakeholders' viewpoinst in a broader sense. Another issue of interest is the proper amount of measures and indicators to implement in an organization to optimizing the two sides of the cost of quality (cost of quality and cost of non quality). How can multiple concurrent control mechanisms across several dimensions of analysis be balanced? The approach of Balancing Multiple Perspectives (BMP) has been designed to help project managers choose a set of project indicators from several concurrent viewpoints. This paper presents the results from a second BMP application in Turkey, using a list of 14 candidate measure. Lessons learned are presented for improving measurement plans.

Successes and challenges experienced in implementing a measurement program in small software organizations

Sylvie Trudel, Pascale Tardif

Centre de Recherche Informatique de Montréal (CRIM), Montréal, Canada

Sylvie.Trudel@crim.ca, Pascale.Tardif@crim.ca

Abstract. In recent years, the authors have implemented measurement programs in several organizations of different sizes. Two of them were small software companies of approximately 12 employees, which were mostly developers. Although these two organizations were similar in size and technology, the differences in the issues they were facing led to completely different approaches for their measurement program. This paper is about the steps taken to implement these measurement programs, both including functional size measurement with COSMIC, effort, schedule, and defects. It also describes what was done to ensure the success of each program and, most importantly, the challenges that were faced during their implementation and maintenance, as well as some of the solutions proposed to answer these challenges.

Organizational Software Measurement Process

Josyleuda M.M. de Oliveira, Karson B. de Oliveira, Arnaldo Dias Belchior

UNIFOR, University of Fortaleza, Fortaleza, Brazil

josymmo@hotmail.com, karlson.oliveira@gmail.com, belchior@unifor.br

Abstract. Software development is a complex activity which demands a series of factors to be controlled. In order for this to be controlled in an effective manner by project management, it is necessary to use software process measurement to identify problems and to consider improvements. This paper presents an organizational software measurement process resulting from the mapping of five relevant software measurement processes: CMMI-SW, ISO/IEC 15939, IEEE Std 1061, Six Sigma, and PSM (Practical Software Measurement). A website was doing to support this process and it helps all the staff understand and use the process. Moreover, the tools should support the successive phases of the measurement process and help maintain the information, because all data will be in the same place and their access is optimized. Thus, the use of the measurement process becomes very easy.

Analysis of Requirement Specifications in Student Projects: A Empirical Study

Michael Olschimke[1], *Cornelius Wille*[1], *Reiner R. Dumke*[2]

[1]Fachhochschule Bingen, Germany

[olschimke|wille]@fh-bingen.de

[2]Otto-von-Guericke-Universität Magdeburg, Germany

dumke@ivs.cs.uni-magdeburg.de

Abstract. Requirement specification is one of the first phases of software development. Software requirements describe the needs and scope of a software product which has to fix a real-world problem. This is also true for student projects at the University of Applied Sciences Bingen. Students as well as software developers in the industry should be able to define their planned software using a software requirements specification.

This empirical study measure and analyze the quality of student specifications in order to improve the quality of these documents and further to improve education at the University of Applied Sciences Bingen.

Based on automated textual analysis of the specification documents we have searched for quality indicators like imperatives or option and weak phrases that normally result in a specific quality of the specification. Then, we have compared the result of the analysis with the individual grade for the student's project in order to find out if there are coherences between the quality of requirements specifications and the student's overall grade for the project. The main goal of our study is to get indices for improving the quality of our courses.

A case study of metric-based and scenario-driven black-box testing for SAP projects

Maya Daneva[1], *Alain Abran*[2], *Olga Ormandjieva*[3], *Manar Abu Talib*[3]

[1]University of Twente

m.daneva@utwente.nl

[2]Uiversité du Québec à Montreal

alain.abran@etsmtl.ca

[3]Concordia University

ormandj@cse.concordia.ca, m_abutal@cse.concordia.ca

Abstract. Enterprise Resource Planning (ERP) projects are perceived as mission-critical initiatives in many organizations. They are parts of business transformation programs and are instrumental in improving organizational performance. In ERP implementations, testing is an activity that is crucial in order to ensure that the functionality embedded in the solution matches the business users' requirements. However, little is known about how to make the testing process more predictable or how to increase its chances of success.
This paper makes a first attempt towards improving the quality of the testing process in ERP projects by using a metric-based test case selection approach. The paper reports on how this approach was adapted to an ERP package-specific project context, how it was applied in five settings in a mid-sized project and what was learnt about using it.

Measuring the Quality Of Inferred Interfaces

Florian Forster

Department of Computer Science, University of Hagen

florian.forster@fernuni-hagen.de

Abstract. Introducing interfaces to a program serves to decouple the code and to increase its flexibility. Type inference algorithms can be used to extract the interface required from an existing type as expressed by a declaration element typed with this type. However, if many variables in a program are typed with the same type, many new interfaces are likely to be deduced these algorithms. Unfortunately, the developer has to trust his intuition deciding whether the new interfaces proposed by the type inference algorithm are worth the trouble, i.e. if the increased decoupling outweighs the additional maintenance effort which comes along with every new interface and vice versa. Therefore, we provide a measurement to compare sets of inferred interfaces with each other, thus helping developers to select the best set of interfaces for his needs. Furthermore, we briefly evaluate our metric and provide a short sketch for the integration of the metric to the Eclipse IDE.

DASMA DIPLOMARBEITEN-PREIS:

Conception and Prototypical Implementation of a Web Service as an empirical-based Consulting about Java Technologies

Ayaz Farooq

University of Magdeburg, Faculty of Computer Science, Magdeburg, Germany

farooq@ivs.cs.uni-magdeburg.de

Zusammenfassung: Die Relevanz von Computer-Systemen im täglichen Leben ist unumstritten. An die dabei zur Anwendung kommende Software steigen die Anforderungen hinsichtlich ihrer Qualität immens. Das Software Engineering will gerade hierbei durch die Anwendung ingenieurtechnischer Maßnahmen, wie zum Beispiel dem Messen und Bewerten einen besonderen Beitrag leisten. Hinsichtlich der Technologie ist heute die Objektorientierung dabei die vorherrschende. Insbesondere nehmen die in Java entwickelten Systeme anteilig deutlich zu.
In den vergangenen Jahren hat aber auch die Rolle der Software-Messung in seiner Form der erfolgreichen Anwendung von Metriken ständig zugenommen. Das kommt auch in der Ausprägung der Prozessbewertungsstufen nach dem Capability Maturity Modell (CMMI) der Stufe 4 als quantitatives Management zum Ausdruck.

DASMA DIPLOMARBEITEN-PREIS:

Zusammenfassung – Design und Implementierung eines anpassbaren Metric Plug-ins für Eclipse

(engl. „Design and Implementation of a customizable metrics plug-in in Eclipse ")

Ansgar Lamersdorf

Zusammenfassung: Die Messung von (Software-) Metriken ist ein essentielles Mittel zur frühzeitigen Vorhersage und Steuerung der Qualität eines Software Produktes. Wichtige nichtfunktionale Eigenschaften wie Zuverlässigkeit (Reliability) und Wartbarkeit (Changeability), die eigentlich nur nach Fertigstellung der Software am konkreten Produkt gemessen werden können, können durch die Erfassung geeigneter Metriken (z.B. über Komplexität oder Größe) schon in früheren Phasen abgeschätzt und vorhergesagt werden.
Ziel dieser Bachelorarbeit war die Entwicklung eines Plug-ins für die Softwareentwicklungsumgebung Eclipse, welches die Messung und Visualisierung von Metriken über die statische Struktur eines Software Produktes unterstützt.

Using COSMIC-FFP for sizing, estimating and Planning in an ERP environment

Frank Vogelezang

Sogeti Nederland B.V.

frank.vogelezang@sogeti.nl

Abstract. Triggered by new European legislation the Dutch Office for Regulations decided to renew major parts of their IT landscape with Oracle's E-Business Suite. They expect that this packaged solution offers the possibility of quick implementation of new business processes.
For the implementation of new regulations and the redesign of existing ones, a software factory was set up with three production lines implementing processchains. Because of the nature of the documentation COSMIC-FFP was used to size the process-chains to be implemented. The measured functional size was used to support the cost estimation and the planning process.
This experience shows that COSMIC-FFP can be used to size, estimate and plan an ERP implementation with a high degree of parameterisation. Since this kind of implementation differs in a number of ways from an average implementation of packaged software future research is necessary.

Mapping Concepts of Functional Size Measurement Methods

Pinar Efe[1], *Onur Demirors*[2], *Cigdem Gencel*[2]

[1]Siemens PSE Turkey - ODTU Teknokent Silikon Bina ZK 21 Ankara, Turkey;

pinar.efe@siemens.com

[2]Informatics Institute, Middle East Technical University - Ankara, Turkey

demirors@ii.metu.edu.tr, cgencel@ii.metu.edu.tr

Abstract. Today, there are many variants of Functional Size Measurement (FSM) methods in use. These methods measure the software using their own concepts and measurement processes and utilize different metrics. Therefore, a piece of software has several functional sizes when measured by different methods. On the other hand, FSM methods share some common concepts and uses related attributes in their measurement processes.
In this paper, common concepts and common measurement possibilities are investigated for three ISO certified FSM methods, which are IFPUG FPA, Mark II FPA and COSMIC FFP. In the light of the findings on the common measurement concepts and rules, a unification model for these methods is proposed in order to measure the software systems using the same source of data. A case study is implemented to an industrial project in order to evaluate this model.

KEYNOTE:

Combat Resistance to Software Measurement by Targeting Management Expectations

Carol Dekkers

Quality Plus Technologies

Abstract. The software Industry has been slow to embrace measurement practices even when software managers recognize the benefits is can deliver. This presentation addresses the issue of resistance and issues related to successful software measurement by addressing management expectations. It includes a discussion of the human and technical factors that are critical to software measurement success.

Measuring the Qualities of Software Design

Naji Habra, Benoît Vanderose

University of Namur – FUNDP, Namur, Belgium

nha@info.fundp.ac.be, bva@info.fundp.ac.be

Abstract. Many software quality models & classical software measurement methods are based on the view according to which the software is one product (sometimes identified more or less to the "code"). A closer view to the software as a product to be measured shows that it is a composed entity made up of several interconnected artifacts (requirement, design, code…). This work starts with this large view of software to propose a first classification of the different software attributes according to the underlying artifacts and their potential relationships. A particular focus is put on the software attributes related to the design artifact.

Error Propagation in Software Measurement and Estimation

Luca Santillo

Independent Consultant, Italian Software Metrics Association Board of Directors

luca.santillo@gmail.com

Abstract. Generically speaking, software measurement and estimation require the application of an algorithm to one or more input variables (measures), in order to provide one or more output variables (estimates, or metrics) for effort, cost, time, quality or other aspects of the software being developed. Regardless of the estimation model (algorithm) being used, practitioners must face the uncertainty aspects of such process: errors in initial measures do affect the derived metrics (or estimated values for indirect variables). Measurement theory does provide an accurate way to evaluate such “error propagation” for algorithmic derivation of variable values from direct measures. Although some software estimation models already propose confidence ranges on their results, the formal application of error propagation can yield some surprising results, depending on the mathematical functional form underlying the model being examined. This work introduces error propagation in the software measurement field and shows some application and examples based on some of the most common software measurement methods and estimation models, as Function Point analysis (for size), Constructive Cost Model (for effort and/pr duration), and others. Proposed cases and examples stimulates critical analysis of methods and models being examined from a possibly new perspective, with regards to the accuracy they can offer in practice.

Generic Metric Extraction Framework

El Hachemi Alikacem[1], *Houari A. Sahraoui*[2]

[1]Centre de Recherche Informatique de Montréal, Québec, Canada

alikacem.el-hachemi@crim.ca

[2]Département d'informatique et de recherche opérationnelle
Université de Montréal, Québec, Canada

sahraouh@iro.umontreal.ca

Abstract. Nowadays, a large number of extraction tools are available. However, using them, it is often difficult to gather and incorporate new metrics. On the other hand, the metric specifications often lack precision and therefore lead to multiple implementation patterns. In this paper, we propose a new approach of metric gathering. This approach, which is at the same time generic and practical, is based on a metric description mechanism. It uses a language that makes it possible to manipulate data from the source code representation model. In a first phase, we have defined a generic model for object oriented code representation. A second phase defines a description language that offers the syntactic constructions necessary for data manipulation of the generic mode.

Market Entry Decisions: Numbers or Politics?

Hans Sassenburg

SE-CURE AG (www.se-cure.ch), CH-3775 Lenk, Switzerland

hsassenburg@se-cure.ch

Abstract. In unpredictable software manufacturer organizations, it is difficult to determine when a software product will be released, the features the product will have, the associated development costs or the resulting product quality. The NPVI-method is presented, enabling a software manufacturer to compare and evaluate different release or market entry strategies. However, information has its price in time and cost, forcing decision-makers to make a trade-off between search costs and opportunity costs. In addition, decision-makers simplify the real world, as they cannot escape the diverse psychological forces that influence individual behaviour. Combined with the potential presence of sources of conflict, this often leads to the situation where different stakeholders experience difference aspiration levels. As such, satisficing behaviour where decision-makers try to find consensus and choose a satisfactory release alternative is a good characterisation of the software release decision-making process as found in practice. Successful adoption of the NPVI-method requires that software manufacturers reach the zone of cost effectiveness for the perfection of information; a zone where numbers make business sense, and can be convincingly used to support informed decision-making.

ESOMIC – Automated effort estimation based on UML specification or source code for object oriented programming languages

Daniel Germanus, Lukas Mrokon

Darmstadt University of Technology

daniel.germanus@gmail.com, lukas.mrokon@googlemail.com

Abstract. This paper focuses on the UML and source code representation of modern objectoriented programming languages in an independent metamodel. Mapping identical concepts in different languages isomorphically enables to write a single metric for a bunch of programming languages. An important aspect was to support UML, so metrics can be run on both source code and formal specifications. Transforming specifications into a query friendly model allows the implementation of methods for effort estimation. Finally, the goal was to automatize any of these methods and to evaluate their significance. Aforementioned functionality is part of the open and extensible software system ESOMIC, the effort estimation and software metrics intelligence center. ESOMIC can be extended to support more object-oriented programming languages, software metrics, and high level methods.

Estimating the effects of project risks in software development projects

Klaus Jantzen[1]*, Gillian Adens*[2]*, Robert Armstrong*[2]

[1]K+K Jantzen Software Services GmbH, D-71116 Gärtringen

klausj@jantzen-software.de

[2]Tassc Limited, Livingston, Scotland

{gillian|robert}@tassc-solutions.com

Abstract. Every software project is exposed to adverse external influences, the so called project risks, that affect the cost and the duration of the project and, possibly, the quality of the products. With a risk analysis it can be determined for a specific project what the risks are. These risks then should be included in a systematic and formal manner in the project estimate in order to obtain a realistic and reliable project estimate and a realistic project plan. We will discuss the way that the project risks are accounted for in currently used estimation methods and we will show a method that is used by a modern estimation tool and which takes the two major properties of the project risks– namely probability of occurrence and impact on the project – into account when calculating a project estimate. Finally we will discuss how risk analysis and risk assessment fit into modern development processes and into CMMI.

Status report on functional size measurement for cross-organizational ERP solutions: problems and alternative approaches

Maya Daneva

University of Twente

m.daneva@utwente.nl

Abstract. Measurement is a fundamental part of any managed activity and functional size of software is the core to successful management of any software work of any magnitude. It is

crucial for estimating project team efforts and normalizing quality attributes such as defect rates, defect density, speed of delivery, and project duration. This paper discusses aspects of the measurement challenge in the context of cross-organizational implementation of large business information systems, namely Enterprise Resource Planning (ERP) systems. We make an account of observations in ERP functional size measurement practice and literature, identify aspects of the gap between practice and research, and report on a recent research initiative at the University of Twente that we plan to carry out with strong industrial participation.

Survey of Automation Tools Supporting COSMIC-FFP – ISO 19761

Anabel Stambollian[1], *Alain Abran*[2]

[1, 2]École de Technologie Supérieure-ÉTS, 1100 Notre-Dame Ouest, Montréal (Québec) Canada H3C 1K3

[1]**anabel.stambollian.1@ens.etsmtl.ca**, [2]**alain.abran@etsmtl.ca**

Abstract. Many software tools have been developed to support the implementation of the ISO-19761 COSMIC-FFP standard on functional size measurement. This paper presents a reference framework made up of the set of functions that is of interest to practitioners who implement ISO functional size measurement standards. It also includes a 2006 survey of COSMIC-related tools available both on the market and in the research community. Finally, a gap analysis is presented in which the functions that still need to be addressed by tool vendors are identified.

Durchführung eines Messprogramms: ein Erfahrungsbericht

Andreas Kowitz[1], *Christian Ofer*[2]

[1]BMW AG

Andreas.Kowitz@BMW.de

[2]3D Systems Engineering GmbH

C.Ofer@3DSE.de

Abstract. Die stetig ansteigenden Elektrik/Elektronik (E/E) Umfänge sind entscheidend für den Innovationsanteil im Automobil. Die Beherrschung der daraus resultierenden Komplexität

ist eine wesentliche Voraussetzung für die Wettbewerbsfähigkeit eines Automobilherstellers oder -zulieferers. Um die zunehmende Komplexität der E/E-Anteile zu beherrschen, hat die BMW Group 2001 ein an CMMI orientiertes Change Programm gestartet. Zentrale Ziele waren dabei u.a. die Stabilisierung des E/E-Entwicklungsprozesses sowie die Vermeidung von Risiken. Wesentlicher Bestandteil des Programms war dabei auch die Einführung eines Metriksystems.

Design of an Integrated Measurement Database for Telecom Systems Development

Martin Kunz[1], *Marek Leszak*[2], *René Braungarten*[1], *Reiner R. Dumke*[1]

[1]Software Engineering Group, University of Magdeburg, Germany

{makunz, braungar, dumke}@ivs.cs.uni-magdeburg.de

[2]Lucent Technologies Network Systems GmbH, Nuernberg, Germany

mleszak@lucent.com

Abstract. The importance of software metrics gathered by measuring artefacts emerging during the software development process for economic and scientific purposes is beyond controversy these days. To help estimate project characteristics, measure project progress and performance or quantify product attributes, and thus to benefit from it in the long run, a suitable defined set of metrics data need to be defined, collected and analysed. In a complex enterprise with large-scale development projects, a structured and persistent central storage solution is almost compulsory. In addition, important statistical techniques for data analysis and visualization techniques are also one major requirement. As an additional target the application of such measurement database facilitates to reach CMMI^{SM} (Capability Maturity Model® Integration) level 3 for all development units, implying the fulfilment of “Measurement and Analysis” Process Area requirements, which contains the Specific Practices such as “Specify Data Collection and Storage Procedures” and “Store Data and Results”, etc. This paper presents results from empirical investigations of the Metrics situation within different departments of Lucent TXS Nuremberg, where we focused on the three major disciplines - System Engineering, Software Development, and System Test. Based on the research of diverse metric data and repositories, the high-level design of a measurement repository based on the Goal-Question-Indicator-Measurement (GQ[I]M) methodology and the CMMI^{SM} framework is presented.

Structuring Software Process Metrics – A holistic semantic network based overview

Reiner R. Dumke[1], *René Braungarten*[1], *Martina Blazey*[2], *Heike Hegewald*[3], *Daniel Reitz*[4], *Karsten Richter*[5]

[1]University Magdeburg, Faculty of Computer Science, Germany

dumke@ivs.cs.uni-magdeburg.de

[2]VW Wolfsburg, Germany, [3]CSC Wonsheim, Germany

[4]EZ T-Systems Berlin, Germany, [5]Bosch Stuttgart, Germany

Abstract. The following paper characterizes the area of software processes considering their different approaches for evaluation and measurement. It shows some of the existing kinds of evaluation (rules of thumb, laws, principles, formulas etc.) and metrics concepts in the software management literature background.
The goal is to identify process quality rules that cover the whole software process models and structures in order to achieve a quantitative software management and to identify open problems. We discuss a methodology achieving a holistic overview about quality-based relations between different components of the software development considering products, processes and resources.

How do we apply statistical process control in the area of software development? – Experiences from industry

Melanie Ruhe

SIEMENS AG, CT SE3, Munich, Germany

melanie.ruhe@siemens.com

Abstract. Applying statistical process control (SPC) is well-known and established in production processes but rather controversial in the area of software (SW) development. The paper provides the description of our framework process for introducing SPC in a development organization of the SIEMENS AG along with some explaining examples. On the other hand the paper states problems and questions that come along with SPC in the area of SW development.
The challenges and experiences are presented and discussed on a detailed level. From our experiences it can be said that the information gained from the practice examples have been essential for decision making in time as well as beneficial process performance predictions. SPC is a helpful tool regarding levers for optimizing processes in mature organizations with repeatable processes & projects. It allows consistent prediction and fine granular monitoring of project performance and thus supports reducing development costs.

Use Case Points in der industriellen Praxis

Stephan Frohnhoff, Volker Jung, Gregor Engels

sd&m AG, Berliner Str. 76, D-63065 Offenbach

frohnhoff@sdm.de

Abstract. Fast and precise effort estimation of software development projects is crucial in IT industry. Within a case study the Use Case Point method has been applied to 10 commercial software development projects and compared with the incurred project efforts after project close. The method is ready for use in commercial projects. We propose appropriate improvements of the Use Case Point method leading to sig-nificantly higher estimation accuracy.

When use COSMIC FFP? When use IFPUG FPA? A Six Sigma View

Dr. Thomas Fehlmann

Euro Project Office AG, Zurich, Switzerland

thomas.fehlmann@e-p-o.com

Abstract. Six Sigma has become a major drive in industry and is rapidly gaining interest in software development and maintenance as well. The Six Sigma management strategy focuses on measurements for reducing defects early in the value chain processes and thus functional sizing measurements are a must for all Six Sigma Green and Black Belts that dare to deal with IT processes, be it in development or operations. However, which measurement method suits better to Six Sigma, the well established IFPUG 4.2 Function Points Analysis, or the more modern ISO standard ISO/IEC 19761, known as COSMIC FFP V2.2?
Interestingly, both measurement methods seem rather complimentary than competing when used in a Six Sigma setting, a setting rather targeted for defect avoidance than for project estimation with commercial or engineering background. The two methods serve different purposes.

KEYNOTE:

The investment in Software Process Improvement (SPI) is this the benefit!

Ton Dekkers

Shell Information Technology International B.V.

Abstract. The investinent in SPI should result in the area of better performance, higher custorner satisfaction, less defects, more reliable agreements and last but not least less cost.
Determination of performance must be implemented in a way that it can be measured before and after the institutionalised SPI. In a real benchmark is also looked at external parties to have insight in the own performance and the potential of the possible improvements.

With the repository (release 9) of the International Software Benchmarking Standards Group (ISBSG), a public open dataset wich over 3000 Software development projects is available to compare the own peiformance. The ISBSG dataset shows the potential results of a more Professional process.
In the presentation, the process and conditions for benehmarking, the possibilities of using the ISBSG data and ltow it can be used as a reference to the results of the implemented SPI. The ISBSG questionnaire provides every organisation a base set of metrics to perform a benchmark. To define additional and more specific rnetrics, the Goal-Question-Metric Method is very effective.
The business case of the investment in a software development tool is used as an example. Based op based an (available) benchmark data the potential improvement of the SPI is quantified and the expectations are made more realistic. Especially the validation of the result of the SPI effort promised by the supplier proved to be very relevant.

BSOA06 – Workshopbericht

1. Motivation zur BSOA-Initiative

Nach Aussage führender IT-Analysten, wie z.B. der Gartner-Group, wird die Etablierung serviceorientierter Architekturen (kurz SOA) die Vorgehensweise bei der Entwicklung neuer Anwendungssysteme in den kommenden Jahren grundlegend beeinflussen. Nicht selten ist in diesem Kontext sogar vom Ende der Dominanz monolithischer Softwarearchitekturen die Rede. Diese Überlegungen beruhen auf der Tatsache, dass bei einer servicebasierten Architektur neue Anforderungen primär auf der Basis bereits existierender fachlicher Serviceangebote realisiert werden können. Die autonom im Netz residierenden Serviceangebote bieten fachlich begründete Funktionen und Daten über eine wohldefinierte Schnittstelle an, kapseln die dahinter liegende Implementierung im Sinne einer „Black Box“ und unterstützen durch eine lose Kopplung die Verwendung innerhalb vielfältiger Anwendungsszenarien. Die eigentliche Intelligenz zur Umsetzung neuer Anforderungen liegt damit in der Komposition von Serviceangeboten und in der endnutzerbezogenen Präsentation. Auf dieser Grundlage sollen redundante Systementwicklungen vermieden und entsprechende Kosten eingespart werden können.

Vor dem Hintergrund eines solchen primär fachlich orientierten Paradigmenwechsels werden neue Bewertungsmodelle benötigt, die prozess-, produkt- und ressourcenbezogene Aspekte im Kontext einer SOA berücksichtigen müssen. Die BSOA-Initiative greift diese Themenstellung auf und beschäftigt sich unter anderem mit den folgenden Aspekten:

- Bewertung der Mehrwertpotenziale einer SOA,
- Erarbeitung von Richtlinien zur Serviceentwicklung für eine SOA,
- Qualitätsbewertung angebotener Services und aufsetzender Kompositionen,
- Mess- und Bewertungsansätze zum Reifegrad einer SOA,
- Services Level Agreements (SLAs) und Verhandlungsaspekte.

Zur bundesweiten Etablierung dieser Initiative wurde am 24.11.2006 ein erster Workshop zum Thema „Bewertungsaspekte serviceorientierter Architekturen“ an der Fachhochschule für Wirtschaft Berlin (Berlin School of Economics) durchgeführt. Etwa 40 Teilnehmer aus allen Teilen Deutschlands und aus Österreich waren zu dem eintägigen Workshop nach Berlin gereist. Das Verhältnis der Teilnehmer aus dem industriellen und akademischen Umfeld hielt sich die Waage. Der Workshop wurde in Kooperation zwischen der FHW Berlin (Fachbereich II – Systementwicklung) und der Otto-von-Guericke-Universität Magdeburg (Softwaremesslabor) unter der Schirmherrschaft der CECMG (Central Europe Computer Measurement Group) veranstaltet und durch die GI (Gesellschaft für Informatik) und die DASMA (Deutschsprachige Interessensgruppe für Softwaremetrik und Aufwandsschätzung) unterstützt.

2. Inhalte des Workshops

Das inhaltliche Interesse galt der Bewertung von IT-gestützten Integrationslösungen (subsumiert unter dem Stichwort SOA), durch die unternehmensinterne, aber auch unternehmensübergreifend genutzte Softwareanwendungen (bzw. zunehmend Serviceangebote) prozessorientiert miteinander verbunden werden können. Aus den

eingereichten Beiträgen wurden im Rahmen eines bundesweit zusammengesetzten Programmkomitees die folgenden, kurz beschriebenen Beiträge ausgewählt. Bei der Auswahl der Themen wurde insbesondere auf einen ausgeglichenen Mix industrieller und wissenschaftlicher Beiträge Wert gelegt. So kamen 3 Vorträge unmittelbar aus der Industrie und 4 Vorträge aus dem universitären Umfeld.

- *Martin Kunz (Otto-von-Guericke-Universität Magdeburg):* Serviceorientierte Ausrichtung von Test- und Messwerkzeugen

  Der Autor geht der Frage nach, wie Funktionalitäten neu zu entwickelnder, aber auch existierender Mess- und Testwerkzeuge als netzwerkbasierte Serviceangebote bereitgestellt werden können. Mit Hilfe einer empirischen Analyse wird sowohl die Sicht des Kunden, als auch die Sicht der Toolhersteller einer ersten Bewertung unterzogen.

- *Heinz-Günter Siebert (Siebert EDV-Beratung & Universität Duisburg-Essen):* Unterstützung von Geschäftsprozessen durch Serviceorientierte Architekturen

  In diesem Beitrag werden serviceorientierte Architekturen (SOA) aus Sicht der Unternehmensführung eingeordnet und bewertet. Die Ausrichtung der eingesetzten Informationssysteme auf die Unternehmensziele steht dabei im Mittelpunkt. Grundprinzipien einer serviceorientierten Architektur werden anhand des SOA-Tempels dargestellt und die mit ihm verbundenen Begriffe durch ein SOA-Ebenen-Glossar zum besseren Verständnis erklärt.

- *Dennis Heinemann (Hochschule Harz & T-Systems Enterprise Services GmbH):* Semantische Aspekte in Service-orientierten Architekturen

  Den innerhalb von serviceorientierten Architekturen verwendeten XML-Nachrichten (typisch SOAP) fehlt zumeist die Zuordnung ihrer eindeutigen Semantik. Mit Hilfe von Informationsobjektmodellen (oder auch Businessobjektmodellen) können semantische Informationen mit Informationsobjekten verknüpft werden. Am Beispiel der Telekommunikationsbrache werden die beiden Modelle SID und BOM kurz vorgestellt und einer ersten Bewertung unterzogen.

- *Matthias Schorer (FIDUCIA IT AG):* Serviceorientierte Architekturen (SOA) — viele Fragen offen

  Die FIDUCIA IT AG als größter IT-Dienstleister für die Volks- und Raiffeisenbanken in Deutschland beschäftigt sich seit langen mit den Möglichkeiten eines servicebasierten Ansatzes. Das Java basierte Banking Framework – JBF, auf dem alle Entwicklungen im Hause FIDUCIA basieren, verfolgt bereits seit der ersten Version im Jahre 1998 einen Service-orientierten Ansatz. Im Zuge des SOA-Hype werden verschiedene marktgängige SOALösungen untersucht und bewertet.

- *Nico Brehm (Universität Oldenburg):* Sicherheitsprotokoll zur Bewertung von Diensten in SOA-basierten Anwendungssytemen

  Die automatische Suche und Entscheidung über die Nutzung von konkurrierenden Diensten ist zumeist mit Sicherheitsproblemen verbunden, da Dienstnutzern oftmals keine Informationen über die Vertrauenswürdigkeit von fremden Dienstanbietern zur Verfügung stehen. Der vorliegende Ansatz beschreibt ein Sicherheitsprotokoll durch dessen Anwendung Bewertungen über Dienste dezentral von den Anbietern selbst verwaltet werden können. Das Sicherheitsprotokoll zielt auf die Verhinderung von Manipulationen durch die Dienstanbieter ab.

- *Dirk Malzahn (OrgaTech Unternehmensberatung Lünen):* Normbasierte Bewertung von SOA-Strukturen

  Die Bewertung einer SOA muss denselben Regeln unterliegen wie die SOA selbst. Der Idealfall wäre daher erreicht, wenn sich die Bewertung einer SOA selbst als Service abbilden ließe. Dieses setzt voraus, dass die Bewertung einer SOA auf Basis anerkannter Standards und Normen erfolgen kann, die über den Fall der Einzelbewertung hinaus, SOAs vergleichbar machen. Zwar gibt es nicht einen einzelnen Standard, der die Bewertung einer SOA in Summe abdeckt, durch eine geschickte Kombination von Standards lassen sich aber alle Aspekte einer SOA durch eine Bewertung abdecken.

- *Dmytro Rud (Otto-von-Guericke-Universität Magdeburg):* Analyse des Performance-Verhaltens von WS-BPEL-Prozessen

  Im Beitrag wird über prototypische Implementierung eines mathematischen Modells zur Analyse des Performance-Verhaltens von mit Hilfe von WS-BPEL (Business Process Execution Language) orchestrierten Geschäftsprozessen berichtet. Die Implementierung des Modells erfolgte auf der Grundlage einer instrumentierten BPEL-Engine, einer Messdatenbank und eines Analyseprogramms.

3. Diskussion und Ausblick

Innerhalb einer moderierten Diskussionsrunde wurden von allen Teilnehmern des Workshops aktuelle Herausforderungen bei der Bewertung serviceorientierter Architekturen identifiziert und erste Vorgehensweisen zur Bearbeitung dieser Themenstellungen im Rahmen der BSOA-Initiative herausgearbeitet. Angeregt wurde diese durch einen einführenden Diskussionsbeitrag zum Thema „Richtlinien und Vorgaben für die Implementierung einer SOA“. Im Folgenden seien einige der identifizierten Herausforderungen im Kontext der Etablierung eines firmenspezifischen und herstellerunabhängigen SOA-Design-Guideline aufgezeigt.

- Herausarbeiten eines geschäftsgetriebenen Ansatzes (Firmenziele berücksichtigen),
- Bereitstellung von Musterlösungen (vgl. Design Pattern bzw. Fallstudien),
- Aufzeigen der signifikanten Unterschiede zur klassischen Applikationsentwicklung,
- Hinweise zu Fragen der Granularität, Skalierung uns Wiederverwendung,
- Unterscheidung zwischen Neu- und Altsystem,
- Vertragliche Gestaltung von Serviceschnittstellen,
- Vorgaben (Empfehlungen?) zu den verwendeten Basistechnologien,
- Bereitstellung zunächst einfacher Regeln (vgl. Rules of Thumb – z.B. Kosten/Nutzen),
- Bereitstellung konkreter Metriken (z.B. Anzahl/Anteil servicebasierter Anwendungen).

Neben diesen Punkten zeigte sich sehr deutlich die unterschiedliche Interpretation der mannigfaltigen Begriffswelt im Kontext einer SOA. Daher gilt es neben der inhaltlichen Ausprägung eines SOA-Design-Guideline die verwendete Begriffswelt klar und widerspruchsfrei im Sinne einer gemeinsam verwendeten Ontologie zu definieren. Darüber hinaus sollten sich die unterschiedlichen Interessen an einer SOA auch in verschiedenen

Rahmenwerken (z.B. Business-, Developer-, Deployment- oder auch Management-Guidelines) wiederfinden. In diesem Zusammenhang zeigten sich auch die ggf. notwendigen organisatorischen Veränderungen (z.B. neue Rollen oder auch neue Berufsorientierungen), hervorgerufen durch die Implementierung einer SOA. Ebenso wird durch eine SOA das Bedürfnis für einen Informationsaustausch zwischen den Beteiligten (Informations-management – Entwicklung/Integration – Betrieb) deutlich höher, als dieses bei klassischen Entwicklungsprojekten der Fall war.

Bis weit nach Ende des offiziellen Workshops hielten die Diskussionen zu den behandelten Themenstellungen an; insbesondere an einem Freitag ein sicheres Indiz für das große Interesse der Teilnehmer an den durch den Workshop aufgegriffenen Aspekten. Im Jahr 2007 wird es weitere Aktivitäten der BSOA-Initiative geben. SOA-Themenstellungen werden im Rahmen der CECMG-Jahrestagung in Nürnberg (Mai 2007) aufgegriffen. Im Herbst wird es den BSOA07-Workshop geben.

4. Publikation

Zur Unterstützung der BSOA-Initiative wurde an der Otto-von-Guericke-Universität Magdeburg ein korrespondierendes Portal (vgl. **http://ivs.cs.uni-magdeburg.de/~gi-bsoa**) eingerichtet. Neben aktuellen Informationen können dort die detaillierte Agenda des Workshops sowie die entsprechenden Präsentationen aller Referenten eingesehen werden. Darüber hinaus findet sich dort auch eine zusammenfassende und ungekürzte Darstellung zu den Inhalten der durchgeführten Diskussionsrunde.

Neben der internetbasierten Publikation der Präsentationen zum Workshop erfolgte auch die Herausgabe eines Tagungsbandes (ISBN 3-929757-95-8) mit den entsprechenden Langfassungen der Beiträge. Exemplare dieses Tagungsbandes wurden in der Bibliothek des Fachbereichs II der FHW Berlin und an der Otto-von-Guericke-Universität Magdeburg hinterlegt. In begrenzter Zahl können diese auch beim Autor dieses Workshopberichts abgerufen werden.

5. Dank

Besonders hervorzuheben sind die perfekten organisatorischen Rahmenbedingungen, die zum Gelingen des Workshops maßgeblich beitrugen und durch Frau Walz, Frau Wenzel und Herrn Kaufmann verantwortet wurden. Ihnen sei an dieser Stelle noch einmal ein herzlicher Dank ausgesprochen. Ebenso geht ein Dank an Herrn Lück für die Gestaltung der Öffentlichkeitsarbeit sowie an Frau Affeldt für die Teilnehmerregistrierung.

Prof. Dr.-Ing. Andreas Schmietendorf

FHW Berlin & CECMG (Ansprechpartner der BSOA-Initiative)

E-Mail: ***andreas.schmietendorf@cecmg.de***

A Critical Analysis of Testing Maturity Model

Ayaz Farooq, Heike Hegewald, Reiner R. Dumke

University of Magdeburg, Institute for Distributed Systems

{farooq, dumke}@ivs.cs.uni-magdeburg.de

Abstract. *Software process establishment, evaluation and improvement are key research areas in the software engineering field today. Testing activities within a software process play a vital role in quality and profitability of the developed product. In this regard, Testing Maturity Model (TMM) is a well known and probably the most comprehensive maturity model for test process assessment and improvement to date. TMM was completed in 1997 and since then it has not been updated. Within the context of latest test issues, advancements in testing techniques & practices and in software process evaluation & improvement, TMM fails to reflect the state-of-the-art of software testing in 2007. This paper critically reviews TMM, mentions its strengths, highlights some of its weaknesses and suggests improvements and future research directions.*

Keywords: *Test process improvement, software process improvement, Testing Maturity Model, TMM, software process*

1 Introduction

In this extremely quality conscious modern software industry, processes, people and technology are believed to play key role in providing quality software products. Software processes is a key research area in the field of software engineering. One implicit assumption in software process research is that improving the software process will improve the software product quality, and better control of the software process will increase project success [10]. Primary issues associated with software processes are process establishment, improvement and evaluation. Some well known general process improvement and evaluation approaches are summarized by Dumke et al. [6][14].

Embedded within the software development process are several other processes such as requirements analysis process, product specification process, design process and testing process [5]. Testing is an important phase in the software development process and is believed to consume major project resources. In this connection, Swinkels [13] investigates available test process improvement (TPI) models. Prominent among them and first of its kind is the Testing Maturity Model (TMM)[1] [4][2][3][5] which was developed to assist software development organizations in evaluating and improving their testing processes.

Since inception of TMM in 1997, new new testing issues have grown, contemporary best testing practices have been developed, and new software process assessment,

[1] TMM (Testing Maturity Model), CMM (Capability Maturity Model), and CMMI (Capability Maturity Model Integration) are all trademarks of their respective owners

evaluation and improvement techniques been introduced. The next sections critically review TMM in view of all these latest advancements and suggest possible future model improvements and research directions.

2 Overview of TMM

The principal inputs to the development of Testing Maturity Model (TMM), as described by its author Ilene Burnstein [5], were CMM V 1.1, Gerlperin and Hetzel's Evolutionary Testing Model [9], survey of industrial testing practices by Durant [7] and Beizer's Progressive Phases of a Tester's Mental Model [1]. TMM consists of a set of five maturity levels, a set of maturity goals and subgoals and associated Activities, Tasks and Responsibilities (ATRs), and an assessment model. This is probably the only available maturity model for the test processes. The model is quite useful from many aspects. Bearing similarity (in principles) with other general process improvement models such as CMM/CMMI and SPICE, this model can easily be integrated into existing process improvement programs of organizations. The assessment process is simple enough to conduct, especially for smaller organizations, and can provide a faster feedback to engineers and management. Burnstein [5] mentions industrial application of this model in several organizations. Olsen and Vinje [12] also found TMM very useful for practical test-planning and post-evaluation of testing process.

3 Critical Review of TMM

Despite the many benefits of the TMM described in section 2 above, there is always a room for improvement in any practice or methodology. In this regard, we critically review this model from four perspectives, i.e. model objective, model construction, assessment process, and model representation.

3.1 Model Objectives

Testing Maturity Model (TMM) was aimed at test process improvement and assessment and at providing best practices and guidelines to test managers, test specialists, and software quality assurance staff to address various testing issues. It was developed about 9 years ago with CMM V 1.1 (1993 release) as a reference model. Since then many new testing issues and techniques have evolved. Research in process evaluation and improvement has brought forth new approaches. With many version changes, CMM itself has now evolved into CMMI for Development V1.2. TMM also needs to be reviewed keeping in view these contemporary developments.

CMM/CMMI was developed based on extensive feedback from research community, government and industry. TMM like CMM/CMMI or other maturity models, is a set of best practices in the field of software testing. TMM's best practices were based mainly on only one industrial survey of testing practices [7] performed 14 years before. Best practices evolve over time. Additionally, TMM did not incorporate change requests from external sources such as industry or test professionals/users

etc. A revision of TMM is imperative to reflect current best testing practices and customer's/professionals' feedback.

Moreover, TMM adopts theoretical style and primarily provides only high level guidelines in the form of ATRs while lacks more elaborate information which could have been provided in the form of examples, typical work products as they are available in case of CMMI, and example deliverables for each step in the testing process. For example, in maturity level 4 which is about management and measurement, ATRs exist for establishing a test measurement program but more specific and practical advice on how to collect, store, analyze and maintain the test measurement data is missing.

3.2 Model Construction

Figure 1 describes three structural components of TMM and two descriptive parts. The figure shows some structural overlapping of different TMM concepts. Part 1 and part 2 of the TMM model derive different elements from the three model components. These parts are not disjoint and overlap since maturity goals are redundantly contained in both part 1 and part 2.

Additionally, internal structure of the maturity levels associates activities, tasks, and responsibilities (ATRs) with maturity subgoals and one expects that he will get a separate set of ATRs to accomplish each of those maturity subgoals. But on the contrary, part 2 of the model organizes ATRs with respect to maturity goals. Either the internal structure should reflect this fact or these ATRs should be defined for each subgoal separately.

It will be worthwhile mentioning that in an earlier review of TMM, Swinkels [13] also observed that TMM did not address issues relating to test environment, reporting, defect management and testware mangement. However, in our opinion, TMM's maturity level 5 is about optimization/defect prevention and addresses the defect management issues.

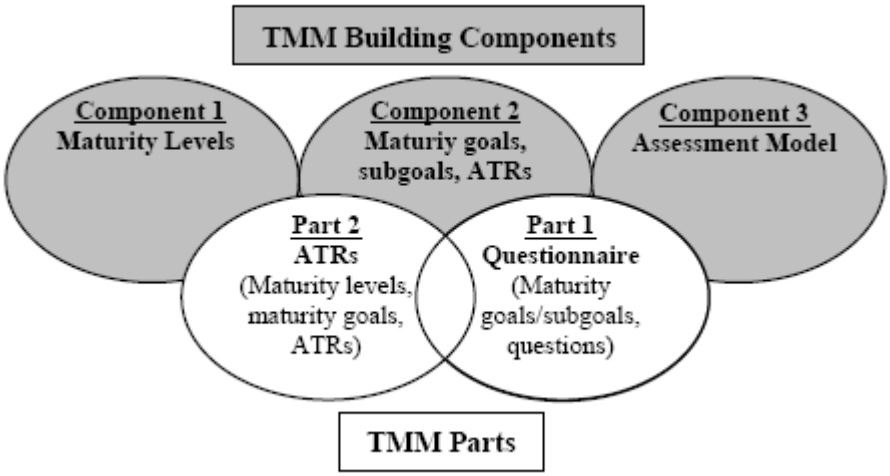


Figure 1: Structure of Testing Maturity Model

TMM ignores people issues as well. Personnel are a key component of processes and their capabilities and involvement are vital to success of the testing process. In a recent empirical study [8], employee participation was identified as the factor with the strongest influence on software process improvement success. Another aspect of people issues is that most of the difficulties in the implementation of improved practices are associated with changing management perceptions, overcoming people's natural resistance to change and implementing workable processes and management controls [13].

3.3 Assessment Process

Different kinds of process assessment approaches exist such as a self, team-based, continuous or independent assessment. Testing Maturity Model provides internal (self) assessment only. This assessment model is not aimed for certification of the testing process by some external body. Assessment and capability determination of a software process are two different activities with different aims. Assessment investigates a process system against a model, standard, or benchmark while a capability determination derives a capability level based on fulfilment of required practices defined in a software engineering process system. TMM's assessment model is lightweight and serves both purposes. However, this kind of internal assessment cannot be used for benchmarking purposes or to compare testing maturity levels among organizations. In today's competitive software industry where CMMI is getting widespread acceptance, provision of a standard (and external) testing maturity determination methodology could testify an organizations testing capabilities.

Moreover, TMM's assessment questionnaire does not advocate using the testing tool questions for maturity level ranking purposes. Usage of appropriate technology/tools is one important aspect for the success of a testing process. In our opinion these testing tool questions could be assembled in more detail and might be a suitable part of assessment and ranking procedures.

3.4 Model Representation

When TMM was developed, CMM supported only a staged representation for process improvement. A staged representation uses predefined sets of process areas to define an improvement path for an organization. Further developments in CMMI introduced a continuous representation as well. The continuous representation enables an organization to select a process area (or group of process areas) and improve processes related to it. While staged and continuous representations have respective pros and cons, the availability of both representations provides maximum flexibility to organizations to address their particular needs at various steps in their improvement programs. In this regard, TMM is inflexible since it supports a staged representation only.

3.5 Related Work

Based on Testing Maturity Model, Jacobs et.al [11] presented (in 2002) basis of a roadmap towards a tentative framework called Metrics based Verification and

Validation Maturity Model. They planned to unite the strengths of known verification & validation improvement models and to reflect proven work practices. It is yet unknown if their proposed model has been completed, is in development, or just abandoned. Another attempt towards improvement of TMM is the establishment of a non-profit organization called TMMi Foundation[2]. This foundation aims to develop a common, robust model of test process assessment/improvement in IT organizations. However, no publicly available document or information yet exists as to the development status of their anticipated model.

4 Future Work

To address the issues raised in this paper about TMM, we are considering a redesign of Testing Maturity Model in line with the newly available version of Capability Maturity Model Integration V 1.2. First, a standard, external, and independent assessment & capability evaluation method will be developed based on CMMI/SPICE. Activities, tasks, and responsibilities (ATRs) can be grouped inside better organized process areas to facilitate the organizations to focus and choose particular improvement areas as per their individual needs. Another proposed extension to this model is provision of both staged and continuous representations. Successful testing depends upon tools/technology, personnel and processes. As mentioned earlier in section 3.2, TMM does not involve all aspects of the software test process maturity such as people, reporting, test tools, and product perspectives. Inclusion of new process areas and a redefinition of maturity levels will also be part of our proposed enhancements to this model.

To cope with the complexity of the testing process and make it more controllable and cost effective and similar to the concept of software development life cycle models such as waterfall, spiral or V-model etc, we are also working on foundations of an adaptive software test process life cycle model incorporating influences of tools/technology, personnel and processes. The proposed life cycle model will derive influences from the IEEE Standard for Developing Software Life Cycle Processes (IEEE Std 1074-1997) and the ISO/IEC 12207 Standard for Information TechnologySoftware life cycle processes.

5 Conclusions

Software test process models, evaluation, assessment and control has attracted attention of many researchers. With its simplicity and ease of assessment and integration, Testing maturity model (TMM) is very useful for test process assessment and improvement. However, in view of current research in process formalization, assessment and improvement, TMM can be improved to adjust some of its drawbacks found in its original version. Furthermore, to cope up with the many issues related to the whole software testing process new concepts such as a software testing process life cycle could bring remarkable changes in this evolving area.

References

[2] http://www.tmmifoundation.org/

[1] B. Beizer: *Software Testing Techniques*. Van Nostrand Reinhold, New York, USA, 1990.

[2] I. Burnstein, T. Suwannasart, and C.R. Carlson: *Developing a testing maturity model: Part 1*. Crosstalk, August 1996.

[3] I. Burnstein, T. Suwannasart, and C.R. Carlson: *Developing a testing maturity model: Part 2*. Crosstalk, September 1996.

[4] I. Burnstein, T. Suwannasart, and R. Carlson: *Developing a testing maturity model for software test process evaluation and improvement*. In Proceedings of the IEEE International Test Conference on Test and Design Validity, pages 581–589, Washington, DC, USA, 1996. IEEE Computer Society.

[5] I. Burnstein: Practical Software *Testing: A Process-oriented Approach*. Springer-Verlag Inc., Secaucus, NJ, USA, 2003.

[6] R.R. Dumke, R. Braungarten, M. Blazey, H. Hegewald, D. Reitz, and K. Richter: *Software process measurement and control – a measurement-based point of view of software processes*. Technical report, Dept. of Computer Science, University of Magdeburg, December 2006.

[7] J. Durant: *Software testing practices survey report*. Technical report, Software Practices Research Center, January 1993.

[8] T. Dyba: *An empirical investigation of the key factors for success in software process improvement*. IEEE Trans. Softw. Eng., 31[5]:410–424, 2005.

[9] D. Gelperin and B. Hetzel: *The growth of software testing*. Communications of the Association of Computing Machinery, 31[6]:687–695, 1988.

[10] M. Huo, H. Zhang, and R. Jeffery: *A systematic approach to process enactment analysis as input to software process improvement or tailoring*. In APSEC'06: Proceedings of the XIII Asia Pacific Software Engineering Conference, volume 0, pages 401–410, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[11] J.C. Jacobs and J.J.M. Trienekens: *Towards a metrics based verification and validation maturity model*. In STEP '02: Proceedings of the 10th International Workshop on Software Technology and Engineering Practice, page 123, Washington, DC, USA, 2002. IEEE Computer Society.

[12] K. Olsen and P.S. Vinje: *Using the testing maturity model in practical test-planning and postevaluation*. In EuroSTAR:98: Proceedings of the 6th European Software Testing, Analysis and Review Conference, pages 345–359, Munich, Germany, 1998.

[13] R. Swinkels: *A comparison of TMM and other test process improvement models*. Technical report, Frits Philips Institute, Technische Universiteit Eindhoven, Netherlands, November 2000.

[14] Y. Wang and G. King: *Software engineering processes: principles and applications*. CRC Press, Inc., Boca Raton, FL, USA, 2000.

Test metrics

Harry M. Sneed

ANECON, Wien / Budapest

As with software development, for the software test metrics serve to evaluate the product, to calculate the project, to measure the progress and to assess the process. So we can distinguish between four types of test metrics – the four “P’s”

- Product metrics
- Project metrics
- Progress metrics and
- Process metrics.

Product metrics:

Let us begin with the product metrics. In testing we are concerned with the testability of the product. A product is testable when it requires a minimum of test cases and test data to demonstrate its correctness and when its results are maximally visible. The goals then are

- To minimize the number of test cases required
- To maximize visibility

Minimizing the number of test cases means different things at different semantic levels. In testing there are at least three such levels

- The unit test level
- The integration test level and
- The system test level.

At the unit test level having less test cases is equivalent to having less paths thru the code to test. The number of logical paths is determined by the number of methods and logical conditions as well as the structure of the classes. Thus, the less methods and conditional branches there are relative to the size of the code in statements, the greater will be the testability. On the other hand, the less levels there are in the class hierarchy relative to the number of classes, the greater will be the testability.

In addition to that, we have data coming in and out of the classes. Much of the input data is just used to set or compute results. However, some of it serves to control the logic in the classes. The more control data we have, the harder it is to test the classes since the control data can not be randomly generated. It has to be explicitly set. Thus, the more control parameters there are the less is the testability.

Taking on these different views of the units under test, we will come up with at least three metrics for unit testability

$$1-\frac{Methods+Branches}{Statements}$$

$$1-\frac{ClassLevels}{Classes}$$

$$1-\frac{ControlParameters}{AllParameters}$$

To demonstrate the measurement of unit testability, assume we have a component of 1500 statements with 7 classes at 3 levels. Each classes has 10 methods each with 5 logical branches. Then, each method has three parameters of which one is a control parameter. That would lead to the following unit testability measures

$$1-\frac{70methods+350branches}{1500statements}=0.72$$

$$1-\frac{3levels}{7classes}=0.57$$

$$1-\frac{70ControlParameters}{210Parameters}=0.67$$

Assuming that all of the metrics are assigned an equal weighting, the average testability of this particular component would be

$$(0.72+0.57+0.67)/3=0.65$$

It's testability could be improved by reducing the number of logical branches, by having less class levels and by decreasing the control parameters.

At the integration test level having less test cases is equivalent to having less interactions between the components and having fewer data interfaces. Interactions between components occur when a method within one component calls a method within another component. These calls are referred to as foreign calls, i.e. long distance calls, as opposed to the local calls of methods within the same component.

Components can also be coupled by sharing the same data. If two separate components access the same database table or use the same file, i.e. one produces the file and the other consumes that file, then they have a data coupling. The relation of the shared database tables and files to the total number of database tables and files is an indicator of testability.

Finally, we have the number of system and user interfaces as opposed to the total number of components. The more interfaces there are the more we have to test.

Having less interfaces reduces the testing burden thus the relation between external interfaces and components is another indicator of integration testability.

From these three views of component integration we derive three metrics for integration testability.

$$1-\frac{ForeignCalls}{AllCalls}$$

$$1-\frac{SharedFiles\ \&\ DatabaseTables}{AllFiles\ \&\ DatabaseTables}$$

$$1-\frac{ExternalInterfaces}{Components}$$

To demonstrate the measurement of integration testability, assume we have 16 components. Within these 16 components there are some 540 method calls of which 96 are foreign calls. These 16 components also process 40 database tables and 10 files. Of the 40 database tables 12 are accessed by two or more components and of the 10 files, 6 are used to pass data between components. The other 4 files are external interfaces. To them come 8 user interfaces giving a total of 12 external interfaces. This results in the following measures of testability

$$1-\frac{96ForeignCalls}{640Calls}=0.85$$

$$1-\frac{18SharedDataStores}{30DataStores}=0.40$$

$$1-\frac{12ExternalInterfaces}{16Components}=0.25$$

Assiming that all of the metrics are assigned an equal weighting, the average integration testability of this particular set of components would be

$$(0.85+0.40+0.25)/3=0.50$$

It is low because of the high number of shared data stores and external interfaces relative to the number of components.

At the system testing level having less test cases is a function of the number of database attributes one has to generate, the number of objects in the user interfaces one has to test and the number of system interfaces one has to deal with.

First, the number of database tables can be viewed in relation to the number of attributes contained within those tables. The more attributes the tables have, the harder it is to populate them. It is easier to deal with many small tables than fewer

larger ones. Since any database table will have at least 2 to 4 attributes as a minimum we must adjust it by at least a factor of 4, thus giving the metric

$$\frac{Tables}{Attributes} \times 4$$

Secondly, systems have user interfaces and user interfaces contain objects or ridgets. Their number drives the effort required to test those user interfaces. The more objects the tester has to manipulate, the higher the test effort. As is the case with the database tables, it is easier to test many user interfaces with few objects than a few interfaces with many objects. Thus, the user interface testability is the relationship of user interfaces to objects contained therein, whereby it is assumed that each user interface has at least two objects, giving the metric

$$\frac{UserInterfaces}{Objects} \times 2$$

Thirdly, there are the system interfaces to deal with. These can be import / export files, remote procedure calls or messages sent and received. Each such interface contains a set of parameters. The number of parameters determines the width of the interface. The more there are, the wider the interface, thus increasing the number of potential data combinations and the required number of test cases. A system with many narrow interfaces requires less effort to test than one with fewer wider interfaces. Thus, we derive the ratio of parameters to interfaces as another measure of system testability. Since an interface will have as a rule at least 3 parameters we must adjust the interface ratio by multiplying it by 3.

$$\frac{SystemInterfaces}{SystemParameters} \times 3$$

To demonstrate the measurement of system testability, let us assume a system has 400 data attributes in 32 tables, that it has 92 widgets or objects in 36 user interfaces and that it has a total of 240 parameters in 24 system interfaces. To measure the testability of this system we use the metrics

$$\frac{32Tables}{400Attributes} \times 4 = 0.08 \times 4 = 0.32$$

$$\frac{36UserInterfaces}{92Objects} \times 2 = 0.39 \times 2 = 0.78$$

$$\frac{24Interfaces}{240Parameters} \times 3 = 0.10 \times 3 = 0.30$$

The average of these three metrics gives us a system testability ratio of 0.46. To obtain a maximum rating of 1, the database tables could have no more than 4 attributes, the user interfaces no more than 2 objects and the system interfaces no more than 3 parameters on average [2].

Project Metrics:

Project metrics are used to estimate the effort and the time required for a test project. As with other cost estimations, the key parameters are the size of the task and the productivity of the workers. Size and productivity in testing are are expressed in terms of the number of test cases required to test a system. This number can be derived from an analysis of the requirements, by counting each action, state and rule as well as each acceptance criteria to be tested. The productivity in test cases per time unit can only be gained through past experience or by copying someone else's experience which is always risky to do.

For estimating test effort and time a modified version of the COCOMO – II method is recommended. The system type and the scaling exponent are the same as in COCOCO – II. [3]. The units of productivity are instead of statements or function–points test cases. The quality adjustment factor is replaced by the testability factor. That results in the equation

$$TestEffort = SystemType \times \left\{ \frac{TestCase^{SE}}{Testproductivity} \right\} \times Testability$$

To demonstrate the use of that equation, we will assume that we have counted 400 test cases and that our previous productivity was 4 test cases per person day or 1 for every 2 hours worked on testing, including test design, test case specification, test data preparation, test execution and test evaluation, but not including test planning. This would give us an unadjusted effort of 100 person days.

Now this has to be adjusted by the testing scaling exponent which ranges from 0.91 to 1.23 depending on five influence factors.

- degree of reuse of previous tests
- testing environment
- target architecture
- test team cohesion
- test process maturity

Each of these factors is evaluated on the scale of 0.91 to 1.23 with 0.91 being the highest fulfillment and 1.23 being the lowest. Then the average is taken. In the case where

- degree of reuse = low = 1.10
- testing environment = medium = 1.00
- target architecture = highly known = 0.96
- team cohesion = low = 1.10
- process maturity = medium = 1.00

we arrive at a scaling exponent of 1.03.

Our raw effort of 100 person days is adjusted by this scaling factor to 115 person days.

Now we must adjust this effort further by multiplying it by the testability factor. To obtain this factor we take the measured system testability of 0.46 which we obtained from the analysis of the data bases, user interfaces and system interfaces. To convert it into a multiplication factor we divide it into the median testability grade of 0.5. A testability ratio higher than the median will reduce the testing effort. A testability ratio lower than the median will increase the testing effort. Here the test effort will be increased by 8% due to the below average testability.

0.5% /0.46 = 1.08 . 115 = 124 person days

The last step is to multiply the adjusted effort by the system type. In COCOMO – II there are 4 system types each with another multiplication factor

Stand alone Application = 0.5

Integrated Application = 1

Distributed Application = 2

Embedded Application = 4

Assuming that we are developing a distributed web application we would multiply the 124 person days by 2 giving a final effort of 248 person days or 12 person months for testing.

To estimate the calender time required we proceed to use the COCOMO – II time equation

$$Time = \left(C \times \underset{PMs}{Effort}^{F} \right) \times \left(1 - SCED\%/100\right)$$

C is a constant depending on the project type. In the case of a new development it is 3.67

F is the time scaling exponent, not to be confused with the effort scaling exponent. It is computed as follows

$$F = D + \left(0.2 \times \left(SE - LB\right)\right)$$

SE is the effort scaling exponent from the effort equation, in our case 1.03. LB is the lower bound of the scaling exponent which is 0.93.

D is a time base coefficient, depending on the project type. For a new development, it is 0.28.

That leaves us with a time scaling factor of

$$(0.28 + (0.2 \times (0.03 - 0.93))) = 0.3$$

The normal time to complete this particular testing project would be

$(3.67 \times 12^{0.3}) = 8$ months using 1 Tester

The SCED% is referred to by Boehm as the schedule compression factor. It is the percentage to which to project time can be reduced by adding more persons to the project. In development it is seldom more than 50% per person, however in testing it can go up to 75% per person. Adding more testers to a testing project will not necessarily make it last longer, provided they are familiar with the project and know what to do.

So for our project assume we can add an additional two testers thus compressing the project time by .75 x .75 = 56% and giving a compression factor of

$$1 - (56/100) = 0.44$$

With two testers we would then need only 8 x 0.44 = 3.5 calendar months. Using the modified COCOMO – II method we have calculated an effort of 12 person months and a duration of 3.5 calendar months for this testing project.

Progress Metrics:

Test progress can be measured in terms of test coverage and errors found. Test coverage is measured in many ways. One is the conventional code coverage, which could be measured at the

method

statement or

branch level

It is expressed as the number of code units traversed by the test relative to the total number of code units, e.g.

$$\frac{BranchesTested}{TotalBranches}$$

Since the advent of frameworks, reuse and code generation this form of coverage has become to mean less. In fact, it has become meaningless unless it is possible to establish a profile of which code units belong to the application under test.

For this reason functional and architectural coverage have become more important. Architectural coverage captures the methods affected by the application, the interactions between those methods and the object types created. It then remains to instrument the code in such a way that these methods, interactions and state transitions are marked. Then the architectural coverage would be

$$\frac{Methods, Calls \,\&\, States Tested}{All\,\mathrm{Re}levantMethods, Calls \,\&\, States}$$

Functional coverage is the easiest to measure provided all of the functions are documented. A static analysis of the functional requirements will tell us how many test cases are required to test all of the functions. The functional test coverage is then

$$\frac{TestCasesExecuted}{TestCases\,\mathrm{Re}quired}$$

Measuring the error detection rate presupposes that the testing team has the error statistics on previous projects or versions, since that is necessary in order to compute the expected error density and to project the number of errors.

There are here two key measurements

- the system size and
- the error count.

The system size can be measured in statements, function points, object points or any other size metric provided it can be extracted from the source code of a system. The error count is the number of errors found by the testers in that particular system. To project the error count on to a future system we need the error density. This is computed by dividing the error count by the system size.

Having completed a system and put it into production the code is analyzed and found to contain 18.000 statements. When testing that system 120 errors were discovered. That gives an error density of

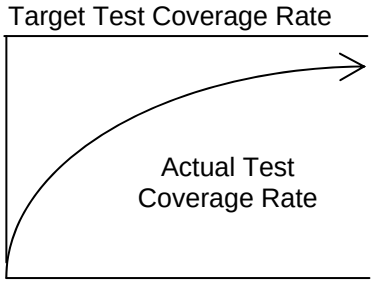
$$\frac{120}{18{,}000} = 0.007$$

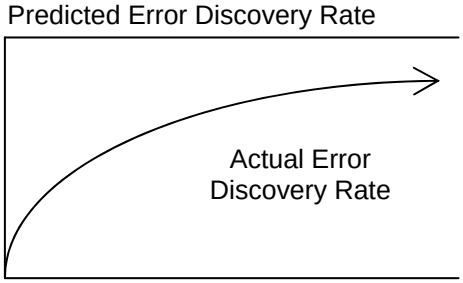
or 7 errors per 1000 statements.

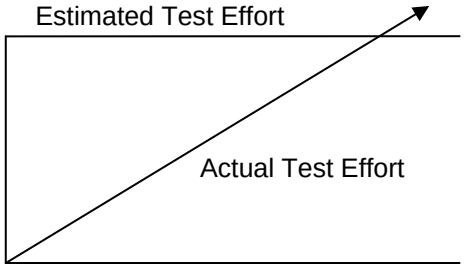
The new system to be developed is estimated to have 1200 Function–Points. From previous measurements we have a ratio of Function–Points to statements in this particular language of 1 to 33. From that ratio we can predict that the new system will have some 39.600 statements. Multiplying that by the previous error density of 0.007 tells us that we should find at least 277 errors in the new system as it is more than 2 times larger than the previous one.

Of course the more systems we have to compare with, the more reliable will be our error projection. The error density rate projected on to the new system could be the median error density of all the comparable systems. In any case, once we have a predicted error count, we can use this as a reference point for the number of errors we should be able to find.

Tracing the test coverage rate and the error discovery rate are the prime means of measuring the progress of a test project. Both metrics can be compared with the rates to be achieved as well as the expended test effort in person days to locate where the project is.







Process Metrics:

In the end every process must be evaluated. To this end we need process metrics to assess the efficiency and effectiveness of the process. Effectiveness is measured in terms of how close the process comes to achieving the goals set for it. Efficiency is measured in terms of the cost incurred in achieving those goals.

The two primary goals of testing are to uncover errors and to instill confidence in the system to be released. [4]. The first goal can be measured by comparing the number of errors uncovered by the testers before the system is released to the number of errors reported by the users after the system is released. The error discovery metric is

$$\frac{Tester\,\mathrm{Re}\,portedErrors}{All\,\mathrm{Re}\,portedErrors}$$

According to the pertinent literature on testing this should be at least 0.85. If in our case 260 errors were found by the testers and another 40 reported by the users, the test efficient would be 0.86 which is sufficient.

The second goal, that achieving confidence, is a function of the errors found in the final test compared to the number of test cases run and the test coverage achieved. The user wants to know that the remaining error probability is very low and that the test coverage is very high. When both factors come together, his confidence in the system is assured. Thus, confidence can be expressed in the following metric

$$TestConfidence = \left\{1 - \frac{ErrorsFoundInLastTest}{TestCasesRun}\right\} \times TestCoverage$$

Assuming that we have run all 400 test cases in the last test and only discovered 3 errors and that the functional test coverage was 95% then the test confidence would be

$$\left\{1 - \frac{3}{400}\right\} \times 0.95 = 0.94$$

The final test metric is the efficiency metric. Efficiency in testing can be expressed in terms of the test cases run and the errors found per person day, smoothed by the achieved test coverage rate.

$$Efficiency = \left\{1 - \frac{Tester\_days}{Errors + TestCases}\right\} \times TestCoverage$$

Assuming we have executed all 400 test cases and found 260 errors with 180 person days of effort and that the final test coverage rate was 0.95 then the test efficiency would be

$$\left\{1 - \frac{180}{260 + 400}\right\} \times 0.95 = 0.69$$

If we would have required 300 person days to run the same number of test cases while only finding 220 errors and achieving a test coverage of only 75%, then the test efficiency would be significantly less.

$$\left\{1 - \frac{300}{220 + 400}\right\} \times 0.75 = 0.39$$

This ends this short discourse on the subject of test metrics. The reader has learned the four main classes of test metrics for measuring

- product testability
- project time and costs
- progress of the test and
- process effectiveness and efficiency.

- In addition, it has been demonstrated how these metrics can be applied. One final remark is that errors are not equivalent. Critical errors weigh more than major errors and major errors weigh more thane minor errors. Therefore, rather than simply counting errors, the reader should consider counting weighted errors. The IEEE Standard 1044 proposes 5 classes of errors

- critical
- severe
- major
- minor and
- disturbing.[5]

The critical errors could be weighted by 8, the severe by 4, the major by 2, the minor by 1, and the disturbing by 0.5. This would help to make the error discovery rate more meaningful in terms of the service rendered. After all, that it what this business is all about – the return on investment – or as Tom DeMarco put it – the bang for your bucks. [6]

Literatur

[1] Sneed, H.: „*Reengineering for Testability*", GI Software-Technik Trends, Band 26, Heft 2, May 2006, p. 8

[2] Sneed, H. / Jungmayr, S.: „*Product and Process Metrics for the Software Test*" InformatikSpektrum, Band 29, Nr. 1, Feb. 2006, p. 23

[3] Boehm, B. a.o.: *Software Cost Estimation with COCOMO – II*, Brentice – Hall, Upper Saddle River, N.J., 1999

[4] Spillner, A. / Linz, I. / Schaefer, H.: *Basic Knowledge of Software Testing*, dpunkt.verlag, Heidelberg, 2006

[5] IEEE 1044: ANSI / IEEE Standard Classification of Software Anomalies, IEEE Computer Society Press, New York, 1993

[6] DeMarco, Tom: *Controlling Software Projects – Management, Measurement & Estimation*, Yourdon Press, New York, 1982

A Formal Representation of Testing Maturity Model (TMM)

Ayaz Farooq, Reiner R. Dumke

University of Magdeburg, Institute for Distributed Systems

{farooq, dumke}@ivs.cs.uni-magdeburg.de

Abstract. *Software process establishment, evaluation and improvement are key research areas in the software engineering field today. Testing activities within a software process play a vital role in quality and profitability of the developed product. Improving the testing process can significantly increase the cost effectiveness of the development process. In this regard, Testing Maturity Model (TMM) is a well known and probably the most comprehensive maturity model for test process assessment and improvement. However, the conventional natural language description of the TMM and other similar process models induces ambiguity, redundancy and inaccuracy in process assessment. To complement the existing descriptive representation of the TMM, this paper introduces a formal presentation of this process model using process algebra and CSP-like notations. This approach can provide us better insight into this process model, and its assessment and capability determination methodology. The developed formal description has also revealed some deficiencies in this model construction and we suggest some changes to improve it.*

Keywords: *Software process, software test process, test process improvement, Testing Maturity Model, TMM, formal process modeling, process modeling language, process algebra*

1 Introduction

In this extremely quality conscious modern software industry, processes, people and technology are believed to play key role in providing quality software products. Software processes is a key research area in the field of software engineering. Overview and general aspects about software process research have been discussed in [1][2][3][4][5]. Primary issues associated with software processes are process establishment, improvement and evaluation. One implicit assumption in software process research is that improving the software process will improve the software product quality, and better control of the software process will increase project success [6]. Software process improvement is probably the most widely discussed issue in the area of software process research. General aspects, techniques, experience reports, and future research directions in software process improvement have been presented in [7][8][9][10][11][12][13][14][15].

Examples of software process improvement models and standards include CMM CMMI, BOOTSTRAP, ISO 9001, ISO 15504 SPICE, ISO IEC 12207 Standard for Information Technology-Software life cycle processes. Dumke [15] and Wang and King [5] present summary and categorization of several software process improvement and capability models. Most of these models are represented in a descriptive style. Available software process improvement (SPI) models lack rigorous and formal description of model structure, process framework, adequacy rating scale,

capability rating scale, and capability determination algorithm [5]. A formal approach, when available, can increase our understandability, reduce ambiguity, and can help us identify model redundancies.

Embedded within the software development process are several other processes such as requirements analysis process, product specification process, design process and testing process [16]. Testing is an important phase in the software development process and is believed to consume major project resources. In this connection, Swinkels [17] investigates available test process improvement (TPI) models. Prominent among them and first of its kind is the Testing Maturity Model (TMM)[1] [18] [19] [20] [16] which was developed to assist software development organizations in evaluating and improving their testing processes. Testing Maturity Model (TMM), like other maturity models and general SPI models, follows descriptive style of representation. In this paper we present a formal description of TMM and investigate how this new approach helps us improve our comprehension of this important model.

2 Modeling Notation

A process modeling language (PML) represents a software process model by using textual, graphical or hybrid notations. A PML allows increased understanding and communication among technical and managerial stakeholders of a software project. Several process modeling languages have been developed until now to cater for different domains and requirements. Many surveys and reports such as [21][22][23][24][25] have classified, assessed and reviewed existing PMLs. For the sake of simplicity and brevity we will use CSP[2]-like process modeling notation/algebra developed by Wang and King [5] to derive a formal representation of the Testing Maturity Model. These authors have already used this algebra to formally describe CMM, ISO 9001, BOOTSTRAP, ISO/IEC 15504 (SPICE) and Software Engineering Process Reference Model (SEPRM). Here we mention only a few essential elements of this notation from [5] which are necessary to understand the following sections.

- **Process**
  - A *process* is defined as a set of activities associated with a set of events $E = \{e_1, e_2 ... e_n\}$, where an event e_i is an internal or external signal, message, variable, scheduling, conditional change, or timing that is specified in association with specific activities in a process.

- **Meta-Processes**
  - *System dispatch* is a meta-process that acts at the top level of a process system for dispatching and/or executing a specific process according to system timing or a predefined event table.

[1] TMM (Testing Maturity Model), CMM (Capability Maturity Model), and CMMI (Capability Maturity Model Integration) are all trademarks of their respective owners
[2] Communicating Sequential Processes

$$SYSTEM \hat{=} \{t_i \Rightarrow P_j \vee e_i \Rightarrow P_j\}, i, j = 1, 2, 3...$$

– *Assignment* is a meta-process that assigns a variable x with a constant value c, i.e:

$$x := c$$

– *Read* or write gets or outs a message from or into a memory location or system port

$$READ \hat{=} l?m \quad or \quad WRITE \hat{=} l!m$$

– *Stop* is a meta-process that terminates a system's operation and is denoted by STOP:

• **Process Relations**

– *Serial* is a process relation in which a number of processes are executed one by one. Assuming two processes are, P and Q, are serial, their relation can be expressed as follows:

$$P; Q$$

– *Pipeline* is a process relation in which a number of processes are interconnected to each other, and a process takes the output of the other process(es) as its input:

$$P \gg Q$$

– The *synchronous parallel* is a process relation in which a set of processes are executed simultaneously according to a common timing system.

$$P \parallel Q$$

3 TMM Process Model

Testing Maturity Model was developed by Ilene Burnstein [18][19][20][16] in 1996/1997 to assist and guide organizations focusing on test process assessment and improvement. The development of TMM was mainly influenced by the then available version of Capability Maturity Model (CMM). Most of the model elements of TMM bear similarity with those of CMM. TMM consists of a set of five maturity levels, a set of maturity goals and subgoals and associated activities, tasks and responsibilities (ATRs), and an assessment model. This is probably the only available maturity model for the test processes. The model is quite useful from many aspects. Owing to its similarity (in principles) with other general process improvement models such as CMM/CMMI and SPICE, this model can easily be integrated into existing process improvement programs of organizations. The assessment process is simple enough to conduct, especially for smaller organizations, and can provide a faster feedback to engineers and management. Burnstein [16] mentions industrial application of this model in several organizations. Olsen and Vinje [26] also found TMM very useful for practical test-planning and post-evaluation of the testing process.

A. Structure of TMM

TMM is structured into certain model elements each of which is only briefly described here.

1) *Maturity Levels:* TMM defines five maturity levels as an evolutionary path to the test process improvement. Figure 1 shows maturity levels of TMM while figure 2 outlines internal structure of the TMM maturity levels. Except for level 1, each level contains a set of maturity goals, supporting maturity subgoals and set of activities, tasks and responsibilities. TMM follows a staged representation of process improvement. A staged representation uses predefined sets of process areas to define an improvement path for an organization.

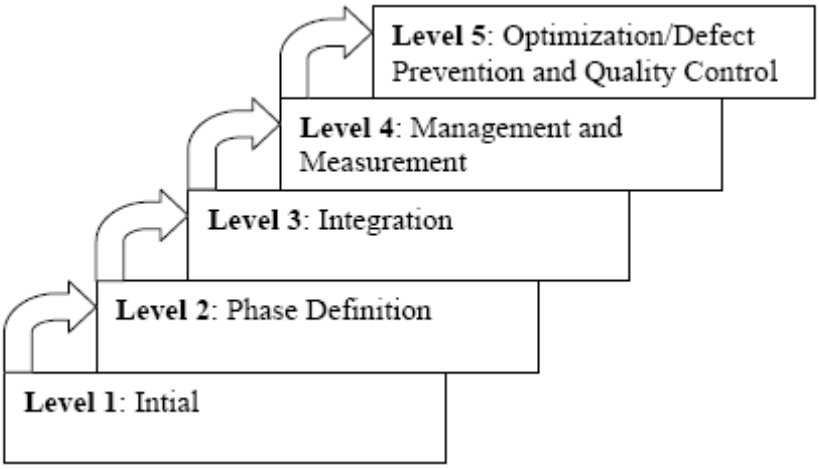


Figure 1: Maturity Levels of TMM

2) *Maturity Goals:* For each maturity level, maturity goals represent testing improvement goals that have to be achieved (in addition to the goals of the preceding level) to satisfy that level. There are 13 maturity goals within TMM.

3) *Maturity Subgoals:* Corresponding to each maturity goal, maturity subgoals exist that outline more concrete steps to be taken to satisfy that goal. There are 43 subgoals altogether in all TMM levels.

4) *Activities, Tasks and Responsibilities:* For each maturity level, a set of even more concrete guidelines are available in the form of activities, tasks and responsibilities (ATRs). They address implementation and organizational adaptation issues at a specific level. ATRs exist for the three critical views related to the testing process, i.e, managers, developers/testers, and users/clients.

B. Framework of the Process Model

Following a brief introduction of TMM model elements above, we now describe framework of the TMM process model with little more detail. Table 1 lists each maturity level, corresponding goals and the main purpose for the existence of that goal.

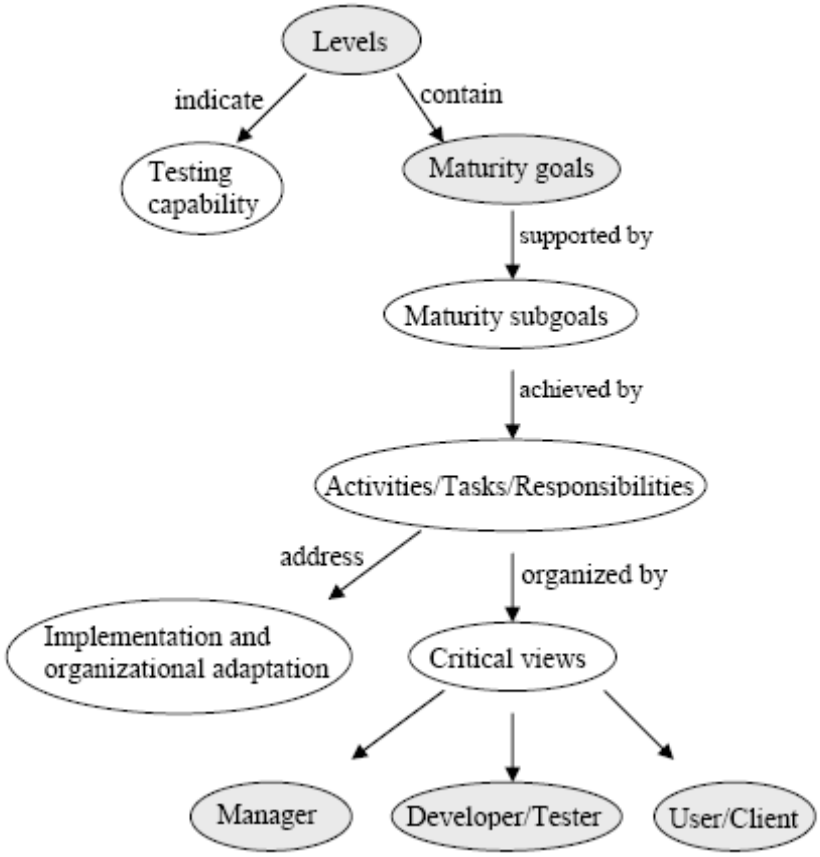


Figure 2: Internal Structure of TMM Maturity Levels

C. Formal Description of TMM

Using the modeling notation/process algebra mentioned in section II earlier, below we formally describe the TMM process model, and interconnection among the goals, subgoals, and ATRs defined within this process model. Such kind of description is very helpful in providing precise definition of TMM structure and interdependence among its modeling elements.

1) *Process Model:* Equation 1 presents a high level view of the TMM process model in a formal manner. Based on our selected notation, we will represent a maturity level with the symbol ML_i where i is the number of the level.

$$\begin{aligned} TMM\_PM \;\hat{=}\; & ML_1 \quad //Initial \\ ; \; & ML_2 \quad //Phase\ Definition \\ ; \; & ML_3 \quad //Integration \\ ; \; & ML_4 \quad //Management\ and \\ & Measurement \\ ; \; & ML_5 \quad //Optimization/Defect \\ & Prevention\ and\ Quality\ Control \end{aligned} \tag{1}$$

2) *Maturity Levels:* Now we formally define the TMM levels. Maturity levels are defined in terms of maturity goals. Maturity goals at a given level are achieved through parallel activities aimed at satisfying these goals. We will represent a maturity goal with the symbol $MG_{i,j}$ where i is maturity level to which this maturity goal corresponds and j stands for the maturity goal number. First maturity level (ML_1) contains no goals and is shown with an empty symbol. Equation 2 through 6 formally define the maturity levels of TMM.

$$ML_1 \;\hat{=}\; \phi \tag{2}$$

$$\begin{aligned} ML_2 \;\hat{=}\; & MG_{2.1} \quad //Scope\ and\ Goals \\ \| \; & MG_{2.2} \quad //Test\ Planning \\ \| \; & MG_{2.3} \quad //Test\ Strategy \end{aligned} \tag{3}$$

$$\begin{aligned} ML_3 \;\hat{=}\; & MG_{3.1} \quad //Organizational\ Process\ Definition \\ \| \; & MG_{3.2} \quad //Training\ Program \\ \| \; & MG_{3.3} \quad //Integrated\ Process\ Management \\ \| \; & MG_{3.3} \quad //Control\ and\ Monitor \end{aligned} \tag{4}$$

$$\begin{aligned} ML_4 \;\hat{=}\; & MG_{4.1} \quad //Reviews \\ \| \; & MG_{4.2} \quad //Measurement \\ \| \; & MG_{4.3} \quad //Quality\ Evaluation \end{aligned} \tag{5}$$

$$\begin{aligned} ML_5 \;\hat{=}\; & MG_{5.1} \quad //Defect\ Prevention \\ \| \; & MG_{5.2} \quad //Quality\ Control \\ \| \; & MG_{5.3} \quad //Continuous\ Improvement \\ & and\ Reuse \end{aligned} \tag{6}$$

3) *Goals:* TMM goals are similar to process areas within CMM/CMMI. Each maturity goal contains two or more subgoals. To avoid lengthy description of numerous equations for each maturity goal, here we give only a generic formal definition of a TMM maturity goal. Although maturity subgoals are not numbered in TMM, we will still refer to a maturity subgoal with the symbol $MSG_{i,j,kj}$, where i represents level number, j the maturity goal, and k_j stands for the total number of maturity subgoals for the jth maturity goal.

$$MG_{i,j} \hat{=} MSG_{i,j,1} \| MSG_{i,j,2} ... MSG_{i,j,k_j} \tag{7}$$

Equation 7 shows that a jth maturity goal at the ith level is satisfied through parallel achievement of k_j maturity subgoals.

Table 1: TMM Process Framework

Maturity Level 2 (ML_2): Phase Definition
Maturity Goals (MGs) and Purposes
$MG_{2,1}$: *Testing goals and scope*
The purpose of this goal is to clearly identify goals, tasks, activities, tools, and policies for the testing and debugging process.
$MG_{2,2}$: *Test planning*
This goal establishes an organizationwide test planning process stating objectives,analyzing risks, outlining strategies, and developing test design specifications, and test cases.
$MG_{2,3}$: *Test strategy*
The purpose of this goal is to identify and apply basic testing techniques and methods to improve test process capability.
Maturity Level 3 (ML_3): Integration
Maturity Goals (MGs) and Purposes
$MG_{3,1}$: *Test Organization*
Purpose of this goal is to identify and organize a group of highly skilled people that is responsible for testing and related issues.
$MG_{3,2}$: *Training Program*
This goal establishes a formal training program based on training policy and specifies training goals and develops specific training plans and materials.
$MG_{3,3}$: *Integrated Process Management*
This goal promotes performing the testing activities in parallel with other life cycle phase activities starting early in the software life cycle.
$MG_{3,4}$: *Control and Monitor*
It is aimed at developing a monitoring and controlling system that can detect deviations from the test plan as soon as possible so that management can take remedial actions immediately.
Maturity Level 4 (ML_4): Management and Measurement
Maturity Goals (MGs) and Purposes
$MG_{4,1}$: *Reviews*
This goal establishes an organizationwide review program to identify, catalog, and remove defects from software artifacts effectively and early in software life cycle.
$MG_{4,2}$: *Measurement*
This goal stresses to identify, collect, analyze, and apply measurements to quantitatively determine quality and effectiveness of various testing activities.
$MG_{4,3}$: *Quality Evaluation*
This step defines and promotes use of measurable software quality attributes, and defines quality goals for evaluating software work products.
Maturity Level 5 (ML_5): Optimization/Defect Prevention and Quality Control
Maturity Goals (MGs) and Purposes
$MG_{5,1}$: *Defect Prevention*
This goal encourages formally classifying, logging, and analyzing defects by using causal defect analysis and action planning.
$MG_{5,2}$: *Quality Control*
Purpose here is to develop a comprehensive set of quality control procedures and practices that support the release of high quality software that fully meets the customer's requirements.
$MG_{5,3}$: *Test process optimization*
The purpose of this maturity goal is to promote continuous test process improvement and test process reuse.

4) *Subgoals:* Although the figure 2 presented earlier shows that maturity subgoals are achieved through set of activities, tasks and responsibilities (ATRs), yet the model description does not give any information as to which maturity subgoals are achieved through which ATRs. On the contrary, ATRs within TMM have been grouped by the maturity goal only. Therefore, any formal equation for maturity subgoals in terms of ATRs cannot be derived. However, here we give formal representation of connection between maturity goals and ATRs in equation 8. In this equation, $ATR - M$ stands for ATRs for managers, $ATR - D$ for ATRs for developers/testers, while $ATR - U$ stands for ATRs for users/clients. $ATR - M_{i,j,lj}$ represents l_j^{th} ATR for managers for the jth maturity goal at the ith maturity level. l_j, m_j , and n_j are different numbers corresponding to total number of ATRs defined (for managers, developers/testers, and users/clients respectively) for the jth goal. It is noteworthy, however, that TMM description does not number ATRs.

$$\begin{aligned} MG_{i,j} &\hat{=} ATR - M_{i,j,1} \| , ... \| ATR - M_{i,j,l_j} \\ &\| ATR - D_{i,j,1} \| , ... \| ATR - D_{i,j,m_j} \\ &\| ATR - U_{i,j,1} \| , ... \| ATR - U_{i,j,n_j} \qquad (8) \end{aligned}$$

Mathematically speaking, the equation 8 gives a multiple definition of maturity goals when compared with equation 7 above. This contradiction in model construction and representation can be removed either by defining ATRs for each maturity subgoal or by changing the maturity level structure diagram (figure 2).

4 TMM Assessment Model

Unlike common process assessment models, the TMM assessment model (TMM-AM) is designed for self-assessment and does not require an external certification body to conduct this process. The TMM assessment model was influenced from CMM and SPICE assessment models. Below we briefly describe its essential components.

A. TMM Assessment Model Components

The three major components of the assessment model are:

1) *Assessment Team Selection and Training:* TMM model provides many guidelines on forming assessment teams and their required abilities, expertise, knowledge, size and other related requirements along with team training advice.

2) *Assessment Procedure:* The assessment procedure comprises certain sequential steps from preparation till implementing improvement. We describe it formally as;

$$
\begin{aligned}
TMM\_Asmt\_Proc \;\hat{=}\; & Preparation \\
; \; & Conducting \\
; \; & Reporting \\
; \; & Analyzing \\
; \; & Action\ Planning \\
; \; & Implementing\ Improvement (9)
\end{aligned}
$$

3) *Assessment Questionnaire:* The TMM assessment questionnaire is the chief, although not the sole, input component for determination of TMM maturity level. In this regard, TMM lacks accurate information on determining the subgoal ranks and leaves it to the discretion of the assessors to calculate these ranks based on questionnaire response alongside additional assessment information gathered through interviews and presentations. On one hand it provides flexibility to the assessors, while on the other hand makes it difficult to precisely describe the ranking procedure in a mathematical and formal manner. Nonetheless, TMM questionnaire contains several maturity goal and subgoal related questions whose response determines fulfillment of those goals and subgoals. Possible answers to these questions have been outlined in table 2.

Table 2: TMM Questionnair Response Set

Response	Practice status
Yes	well established & consistently performed
No	not well established & inconsistently performed
Does not apply	does not apply
Not known	no proper information available

B. Ranking Procedure/Capability Determination

TMM ranking procedure, conducted after an assessment procedure, determines project or organization's testing maturity level.

1) *Rating Scale for Goals/Subgoals:* Table 3 lists the rating scales for the maturity goals and subgoals.

Table 3: Rating Scale for Maturity Goal/Subgoal

Scale	When goal/subgoal is:
Satisfied	implemented and instituted
Unsatisfied	weakly implemented and/or weakly instituted
Not applicable	not relevant
Not rated	beyond scope of appraisal

2) *Degree of Satisfaction for Subgoals:* An additional finer level ranking is available for TMM maturity subgoals mainly aimed at helping to identify strong and weak areas in the testing process. This ranking scale is called *degree of satisfaction*. Degree of satisfaction is determined based on responses to the assessment questionnaire. Table 4 lists degree of satisfaction levels along with guidelines on calculating this ranking.

Table 4: Determining Degree of Ssatisfaction for Maturity Subgoals

Degree of Satisfaction	Questionnaire Response
Very High	% of 'yes' responses is > 90
High	% of 'yes' responses is 70-90
Medium	% of 'yes' responses is 50-69
Low	% of 'yes' responses is 30-49
Very Low	% of 'yes' responses is < 30

3) *Ranking Procedure:* TMM ranking procedure first determines maturity subgoal ranks, then maturity goal ranks and finally maturity levels. Table 5 outlines calculation of maturity subgoal ranks which depend upon responses to the assessment questionnaire. This table says that 50% of the responses to the questionnaire have to be 'yes' to satisfy the relevant maturity subgoal. Therefore, we define a pass threshold, $PThresh_{ques(i,j,k)}$ for maturity subgoals, which is the actual number of questions for each maturity subgoal whose response has to be affirmative. Equation 10 defines this pass threshold.

$$PThresh_{ques(i,j,k)} = N_{ques(i,j,k)} * 50\% \qquad (10)$$

where $N_{ques(i,j,k)}$ stands for the total number of questions for the kth maturity subgoal under the jth maturity goal at the ith maturity level.

TMM guidelines to calculate rank of maturity subgoal have been summarized in table 5. To represent if a maturity subgoal is satisfied or not, equation 11 defines a boolean variable $SATMSG_{(i,j,k)}$ which returns to true when the number of 'yes' responses to the questionnaire is greater or equal to the pass threshold defined above.

Table 5: Calculation of Maturity Subgoal Ranking

Rating Scale	Questionnaire Response
Satisfied	% of 'yes' responses is ≥ 50
Unsatisfied	% of 'yes' responses is < 50
Not applicable	% of 'doesn't apply' responses is ≥ 50
Not rated	% of 'not known' responses is ≥ 50

$$SAT_{MSG(i,j,k)} = true\ if\ P_{ques(i,j,k)} \geq PThresh_{ques(i,j,k)} \quad (11)$$

where $P_{ques(i,j,k)}$ stands for the number of subgoal related questions whose response was affirmative while i, j, k stand for maturity level, goal and subgoal, respectively.

To calculate maturity goal rankings, TMM provides guidelines which have been summarized in table 6. Similar to the maturity subgoal, equation 12 defines a boolean variable, $SAT_{MG(i,j)}$, to represent if a maturity goal is satisfied or not.

Table 6: Calculation of Maturity Goal Ranking

Goal Rank	Requirement on Subgoal Ranking
Satisfied	% of satisfied MSG is ≥ 50
Unsatisfied	% of satisfied MSG is < 50
Not applicable	% of not applicable MSG is ≥ 50
Not rated	% of not rated MSG is ≥ 50

$$SAT_{MG(i,j)} = true\ if\ SAT_{MSG(i,j,k)}, for\ all\ \ k = 1..n_j \quad (12)$$

where n_j stands for the total number of subgoals corresponding to the jth goal at the ith maturity level. This equation shows that a maturity goal is satisfied when all the maturity subgoals for this goal are satisfied too.

Now we define an equation to represent if a maturity level is satisfied. Since TMM is a staged model, to achieve a given maturity level, all lower maturity levels have also to be achieved first. Equation 13 shows that to satisfy a given maturity level all the maturity goals for that level have to be satisfied ($SAT_{MG(i,j)}$ = *true*) and all previous maturity levels have to be satisfied ($SAT_{ML(k)}$ = *true*) as well.

$$\begin{aligned} SAT_{ML(i)} &= true\ if \{SAT_{MG(i,j)}, for\ all\ j = 1..n_i \\ &\wedge\ \ SAT_{ML(k)}\ for\ all\ k = 1, ..i-1\} \end{aligned} \quad (13)$$

where n_i stands for the total number of maturity goals at the ith maturity level while rest of the variables have already been explained above.

Maturity levels may be associated with a particular project or whole organization. Equation 14 determines the project maturity level which is the highest number of maturity level satisfied. TMM descriptions say that if only one representative project was selected for assessment, then the project maturity level also refers to the organization maturity level. In case of more than one assessed project, either all projects will be at the same maturity level or they may also be assessed at different levels. A difference of two levels among project maturities within the same organization is generally considered unusual. However, if about 80% of the projects have been assessed at a given maturity level, that level can be considered to represent organization maturity level as well. In other cases, determination of organization maturity level should consider factors such as number and importance of the projects.

$$ML_{proj} = max\{i|SAT_{ML(i)}, i = 1,..5\} \qquad (14)$$

5 Conclusions and Future Work

With its simplicity and ease of assessment and integration, Testing Maturity Model (TMM) is very useful for test process assessment and improvement. We have applied formal approach to the description of Testing Maturity Model. A selected process algebra and mathematical notations have been adopted for presenting the TMM process model, process assessment model, and maturity level ranking procedures in the form of mathematical equations and expressions. This formal representation have revealed some contradictions in model construction and impreciseness in the assessment model.With some improvements TMM process maturity determination can be automated with little human intervention. Over and above, formal approach ,when applied to general process models or test processes, is a helpful tool to develop a better understanding and implementation of these models.

References

[1] B. Curtis, M.I. Kellner, and J. Over: *Process modeling. Commun*. ACM, 35(9):75–90, 1992.

[2] W. Scacchi: *Process models in software engineering*. J. J. Marciniak (Ed.), Encyclopedia of Software Engineering,, 2002.

[3] A. Fuggetta: *Software process: a roadmap*. In ICSE ’00: Proceedings of the Conference on The Future of Software Engineering, pages 25–34, New York, NY, USA, 2000. ACM Press.

[4] S.T. Acua and X. Ferr: *Software process modelling*. In ISAS-SCI ’01: Proceedings of the World Multiconference on Systemics, Cybernetics and Informatics, pages 237–242. IIIS, 2001.

[5] Y. Wang and G. King: *Software engineering processes: principles and applications*. CRC Press, Inc., Boca Raton, FL, USA, 2000.

[6] M. Huo, H. Zhang, and R. Jeffery: *A systematic approach to process enactment analysis as input to software process improvement or tailoring*. In APSEC’06: Proceedings of the XIII Asia Pacific Software Engineering Conference, volume 0, pages 401–410, Los Alamitos, CA, USA, 2006. IEEE Computer Society.

[7] W.S. Humphrey: *Managing the software process*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.

[8] M. Haug, E.W. Olsen, and L. Bergman: *Software process improvement: metrics, measurement, and process modelling*. Springer-Verlag, Berlin, Germany, 2001.

[9] B. Curtis: *Software process improvement: best practices and lessons learned*. In ICSE ’00: Proceedings of the 22nd international conference on Software engineering, page 828, New York, NY, USA, 2000. ACM Press.

[10] V.R. Basili, F.E. McGarry, R. Pajerski, and M.V. Zelkowitz: *Lessons learned from 25 years of process improvement: the rise and fall of the nasa software engineering laboratory*. In ICSE ’02: Proceedings of the 24th International Conference on Software Engineering, pages 69–79, New York, NY, USA, 2002. ACM Press.

[11] F. O'Hara: *European experiences with software process improvement*. In ICSE '00: Proceedings of the 22nd international conference on Software engineering, pages 635–640, New York, NY, USA, 2000. ACM Press.

[12] B.C. Hardgrave and Deborah J. Armstrong: *Software process improvement: it's a journey, not a destination*. Commun. ACM, 48(11):93–96, 2005.

[13] M.A. Serrano: *State of the art and future of research in software process improvement*. In COMPSAC '04: Proceedings of the 28th Annual International Computer Software and Applications Conference, page 239, Washington, DC, USA, 2004. IEEE Computer Society.

[14] D.N. Card: *Research directions in software process improvement*. In COMPSAC '04: Proceedings of the 28th Annual International Computer Software and Applications Conference, page 238, Washington, DC, USA, 2004. IEEE Computer Society.

[15] R.R. Dumke, R. Braungarten, M. Blazey, H. Hegewald, D. Reitz, and K. Richter: *Software process measurement and control - a measurementbased point of view of software processes*. Technical report, Dept. of Computer Science, University of Magdeburg, December 2006.

[16] I. Burnstein: *Practical Software Testing: A Process-oriented Approach*. Springer-Verlag Inc., Secaucus, NJ, USA, 2003.

[17] R. Swinkels: *A comparison of TMM and other test process improvement models*. Technical report, Frits Philips Institute, Technische Universiteit Eindhoven, Netherlands, November 2000.

[18] I. Burnstein, T. Suwannasart, and R. Carlson: *Developing a testing maturity model for software test process evaluation and improvement*. In Proceedings of the IEEE International Test Conference on Test and Design Validity, pages 581–589, Washington, DC, USA, 1996. IEEE Computer Society.

[19] I. Burnstein, T. Suwannasart, and C.R. Carlson: *Developing a testing maturity model: Part 1*. Crosstalk, August 1996.

[20] I. Burnstein, T. Suwannasart, and C.R. Carlson: *Developing a testing maturity model: Part 2*. Crosstalk, September 1996.

[21] P. Armenise, S. Bandinelli, C. Ghezzi, and A. Morzenti: *A survey and assessment of software process representation formalisms*. Int. Journal on Software Engineering and Knowledge Engineering, 3(3):401–426, 1993.

[22] J.-C. Derniame, B.A. Kaba, and D.G. Wastell: *Software Process: Principles, Methodology, Technology*. Springer-Verlag, London, UK, 1999.

[23] K.Z. Zamli: *Process modeling languages: A literature review*. Malaysian Journal of Computer Science, 14(2):26–37, 2001.

[24] K.Z. Zamli and P.A. Lee: *Taxonomy of process modelling languages*. In AICCSA '01: Proceedings of the ACS/IEEE International Conference on Computer Systems and Applications, page 435, Washington, DC, USA, 2001. IEEE Computer Society.

[25] K.Z. Zamli: *A survey and analysis of process modeling languages*. Malaysian Journal of Computer Science, 17(2):68–89, 2004.

[26] K. Olsen and P.S. Vinje: *Using the testing maturity model in practical test-planning and post-evaluation*. In EuroSTAR:98: Proceedings of the 6th European Software Testing, Analysis and Review Conference, pages 345–359, Munich, Germany, 1998.

Lanza, M.; Marinescu, R.:

Object-Oriented Metrics in Practice

Springer-Verlag Berlin Heidelberg, 2006 (205 Seiten)
ISBN-10 3-540-24429-8
ISBN-13 978-3-540-24429-5

Metrics are paramount in every engineering discipline. However, due to its lack of rigor and its intrinsic complexity, software engineering is not considered a classical engineering activity. Moreover, defining, understanding and applying software metrics often looks like an overly complex activity, recommended only to 'trained professionals'. In general, if a software system is delivering the expected functionality, only few people – if any – care about measuring the quality of its internal structure. Consequently, software metrics are still regarded rather circumspectly by most software developers.

Lanza and Marinescu demystify the design metrics used to assess the size, quality and complexity of object-oriented software systems. Based on a novel approach, backed by generally accepted semantics for metrics and by statistical information from many industrial projects, they deduce a suite of metrics-based patterns for assessing the design of object-oriented software systems. They show in detail how to identify design disharmonies in code, and how to devise and apply remedies.

The combination of theoretically sound results and practically tested procedures and solution paths makes this book an ideal companion for professional software architects, developers and quality engineers. The pattern-oriented description of disharmonies offers easy access to detecting shortcomings and applying solutions to real problems.

Laird, L.M.; Brennan, M.C.:

Software Measurement and Estimation: A Practical Approach

IEEE Computer Society, Wiley Interscience, 2006 (257 Seiten)
ISBN 3-471-67622-5

The text begins with the foundations of measurement, identifies the appropriate metrics, and then focuses on techniques and tools for estimating the effort needed to reach a given level of quality and performance for a software project. All the factors that impact estimations are thoroughly examined, giving you the tools needed to regularly adjust and improve your estimations to complete a project on time, within budget, and at an expected level of quality.

This text includes several features that have proven to be successful in making the material accessible and easy to master:

- Simple, straightforward style and logical presentation and organization enables you to build a solid foundation of theory and techniques to tackle complex estimations

- Examples, provided throughout the text, illustrate how to use theory to solve real-world problems
- Projects, included in each chapter, enable you to apply your newfound knowledge and skills
- Techniques for effective communication of quantitative data help you convey your findings and recommendations to peers and management

Software Measurement and Estimations: A Practical Approach allows practicing software engineers and managers to better estimate, manage, and effectively communicate the plans and progress of their software projects. With its classroom-tested features, this is an excellent textbook for advanced undergraduate-level and graduate students in computer science and software engineering.

Kandt, R.K.:

Software Engineering Quality Practices

Auerbach Publications, 2006 (256 Seiten)
ISBN 3-8493-4633-9

Software Engineering Quality Practices describes how software engineers and the managers who supervise them can develop quality software in an effective, efficient, and professional manner. This volume conveys practical advice quickly and clearly while avoiding the dogma that surrounds the software profession. It concentrates on what the real requirements of a system are, what constitutes an appropriate solution, and how you can ensure that the realized solution fulfils the desired qualities of relevant stakeholders. The book also discusses how successful organizations attract and keep people who are capable of building high-quality systems.

The author succinctly describes the nature and fundamental principles of design and incorporates them into an architectural framework, enabling you to apply the framework to the development of quality software for most applications. The text also analyzes engineering requirements, identifies poor requirements, and demonstrates how bad requirements can be transformed via several important quality practes.

Abran, A.; Bundschuh, M.; Büren, G.; Dumke, R.R.:

Applied Software Measurement

Shaker Verlag, Aachen, November 2006 (542 Seiten)
ISBN 3-8322-5611-3

In this proceedings published at the Deutsche Universitätsverlag and the Shaker-Verlag, Aachen, 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 in Argentina, Australia, Austria, Bahrain, Belgium, Brazil, Bulgaria, Canada, Finland, France, Germany, Ghana, Italy, Netherlands, Slovenia, Spain, Switzerland, UK, USA and Vietnam.

About the contents see the workshop report in this Metrics News.

Büren, G.; Bundschuh, M.; Dumke, R.:

MetriKon 2005 – Praxis der Software-Messung

Shaker Verlag, Aachen, November 2005 (299 Seiten)
ISBN 3-8322-4615-0

The book includes the proceedings of the DASMA Metric Conference **MetriKon 2005** held in Kaiserslautern in November, 2005, 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.

The contents are described by the listing of the paper abstracts in this Metrics News.

Abran, A.; Dumke, R.:

Innovations in Software Measurement

Shaker Verlag, Aachen, September 2005 (456 Seiten)
ISBN 3-8322-4405-0

The book includes the proceedings of the 15th International Workshop on Software Measurement (IWSM2005) held in Montreal in September, 2005, 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 in Argentina, Australia, Austria, Bahrain, Belgium, Brazil, Bulgaria, Canada, Finland, France, Germany, Ghana, Italy, Netherlands, Poland, Slovenia, Spain, Switzerland, UK, USA and Vietnam.

The contents are described by the listing of the paper abstracts in this Metrics News.

Ebert, C.:

Systematisches Requirements Management
Anforderungen ermitteln, spezifizieren, analysieren und verfolgen

dpunkt.verlag, August 2005 (320 Seiten)
ISBN 3-89864-336-0

Projekte scheitern häufig wegen unzureichendem Requirements Management. Meist waren schon zu Beginn die Anforderungen nicht ausreichend geklärt und damit konnte auf deren Änderungen auch nicht richtig reagiert werden. Das Buch bietet einen Überblick über Theorie und Praxis des Requirements Management. Es beschreibt, wie Anforderungen entwickelt, gesammelt, dokumentiert und im Projekt verfolgt werden. Die grundsätzlichen Methoden, Verfahren, Werkzeuge und Notationen des Requirements Management werden übersichtlich behandelt. Sie werden durch konkrete Beispiele aus der Projektarbeit illustriert.

Als Beispiel einer modernen Methode der Anforderungsbeschreibung werden Use-Case-Szenarien in der UML-Notation verwendet. Praktische Fallstudien unterstützen die konkrete Umsetzung.

Leser: Produktmanager, Projektleiter, Softwareentwickler

Weitere Informationen finden Sie unter

http://www.dpunkt.de/buch/3-89864-336-0.html

Sneed, H.M.:

Software-Projektkalkulation – Wissen was Projekte wirklich kosten

Hanser-Verlag, 2005 (228 Seiten)
ISBN 3-446-40005-2

Wer einmal die Kosten oder die Zeit für ein Software-Projekt falsch kalkuliert hat, weiß, dass kein Unternehmen sich das öfter leisten kann. Projektkalkulation ist eine Überlebensfrage der Software-Industrie. Für Auftragnehmer wie für Auftraggeber ist die richtige Kalkulation unabdingbar für den Projekterfolg.

Die meisten Techniken für Aufwandsschätzung, die in der Praxis verbreitet sind, eignen sich nur bei IT-Projekten für eine Neuentwicklung. Geht es in Ihrem Projekt jedoch um Wartung, Migration, Integration oder Sanierung, so müssen Sie darauf abgestimmte Methoden einsetzen. Dieses Buch zeigt, welche Techniken der Aufwandsschätzung für welche Art von Projekten zu nutzen sind.

Preprints/Technical Reports:

Dumke, R.; Schmietendorf, A.; Zuse, H.: *Formal Description of Software Measurement and Evaluation*. University of Magdeburg, 2005

Braungarten, R.; Kunz, M.; Dumke, R.: *An Approach to Classify Software Measurement Storage Facilities*. University of Magdeburg, 2005

Dumke, R.; Braungarten, R.; Blazey, M.; Hegewald, H.; Reitz, D.; Richter, K.: *Software Process Measurement and Control – A Measurement-Based Point of View of Software Processes*, University of Magdeburg, 2006

see as pdf files:

http://ivs.cs.uni-magdeburg.de/sw-eng/agruppe/forschung/Preprints.shtml

WOSP 2007:

5th International Workshop on Software & Performance
February 5-8, 2007, Buenos Aires, Argentina
see: **http://www.wosp-conference.org/**

IASTED SE 2007:

IASTED International Conference on Software Engineering 2007
February 13-15, 2007, Innsbruck, Austria
see: **http://www.iasted.org/conferences/home-552.html**

SEPG 2007:

19th Software Engineering Process Group Conference
March 26-29, 2007, Austin, Texas, USA
see: **http://www.sei.cmu.edu/sepg/2007/**

CSMR 2007:

11th European Conference on Software Maintenance and Reengineering
March 21-23, 2007, Amsterdam, Netherlands
see: **http://www.cs.vu.nl/csmr2007/**

EASE 2007:

10th International Conference on Empirical Assessment in Software Engineering
April 2-3, 2007, Staffordshire, UK
see: **http://ease.cs.keele.ac.uk/**

FSS 2007:

2th Annual Functional Sizing Summit
April 22-26, 2007, Vancouver, Canada
see: **http://www.ifpug.org/conferences/annual.htm**

PSQT 2007:

International Conference on Practical Software Quality & Testing
May 7-11, 2007, Las Vegas, USA
see: **http://www.psqtconference.com/2007west/**

SMEF 2007:

Software Measurement European Forum
May 9-11, 2006, Rome, Italy
see: **http://www.iir-italy.it/smef2007/**

SPICE 2007:

7th International SPICE Conference on Process Assessment and Improvement
May 9-11, 2007, Seoul, Korea
see: http://www.spice2007.com/

ICSE 2007:

International Conference on Software Engineering
May 20-26, 2007, Minneapolis, USA
see: **http://web4.cs.ucl.ac.uk/icse07/index.php?id=75**

ESEPG 2007:

12th European Software Engineering Process Group Conference
June 11-14, 2007, Amsterdam, Netherlands
see: **http://www.espi.org/sepg/**

SIGMetrics 2007:

ACM SIGMetrics - Performance 2007
June 12-16, 2007, San Diego, USA
see: **http://www.cs.cmu.edu/~sigm07/**

ICPC 2007:

15th International Conference on Program Comprehension
June 26-29, 2007, Banff, Canada
see: **http://www.program-comprehension.org/**

PROFES 2007:

8th International Conference on Product Focused Software Process Improvement
July 2-4, 2007, Riga, Latvia
see: **http://www.profes2007.org/**

UKPEW 2007:

21th Annual United Kingdom Workshop on Performance Engineering
July 9-10, 2007, Lancashire, UK
see: **http://www.edgehill.ac.uk/Faculties/HMSAS/Business/UKPEW/**

ICWE 2007:

5th International Conference on Web Engineering
July 16-20, 2007, Como, Italy
see: **http://www.icwe2007.org/**

SPPI 2007:

Software Process and Product Improvement – Euromicro Conference
August 27-31, 2007, Lübeck, Germany
see: **http://em2007.uni-kl.de/cfp_sppi.shtml/**

QFD 2007:

19th Symposium on Quality Function Deployment
September 5-14, 2007, Williamsburg, USA
see: **http://www.qfdi.org/**

PSQT 2007 North:

International Conference on Practical Software Quality & Testing
September 10-14, 2007, Minneapolis, USA
see: **http://www.PSQTConference.com**

QEST 2007:

3rd International Conference on Quantitative Evaluation of SysTems
September 16-19, 2007, Edinburgh, Scotland
see: **http://www.qest.org/qest2007/**

ASQT 2007:

Arbeitskonferenz Softwarequalität und Test 2007
September 20-21, 2007, Klagenfurt, Austria
see: **http://www.asqt.org/**

ESEM 2007:

International Symposium on Empirical Software Engineering & Measurement
September 20-21, 2007, Madrid, Spain
see: **http://www.esem-conferences.org/esem/**

CONQUEST2007:

10. International Conference on Software Quality
September 26-28, 2007, Potsdam, Germany
see: **http://www.conquest-conference.org/**

UKSMA 2007:

18th Annual UKSMA Conference – Managing your Software (through Measurement)
October , 2007, London, UK
see: **http://www.uksma.co.uk/**

QSIC 2007:

International Conference on Software Quality
October 26-28, Beijing, China
see: **http://www.goingtomeet.com/conventions/details/1121/**

IWSM/MENSURA 2007:

17th International Workshop on Software Measurement/2th International Conference on Software Product and Process Measurement
November 5-7, 2007, Mallorca, Spain
see: **http://ivs.cs.uni-magdeburg.de/iwsm2007/**

MetriKon 2007:

DASMA Workshop
November 12-14, 2007, Kaiserslautern, Germany
see: **http://www.metrikon.de**

BSOA2007:

2. Workshop Bewertungsaspekte service-orientierte Architekturen
12. November 2007 in Kaiserslautern,
see: **http://ivs.cs.uni-magdeburg.de/~gi-bsoa/**

see also: **OOIS, ECOOP** and **ESEC** European Conferences

Other Information Sources and Related Topics

- **`http://rbse.jsc.nasa.gov/virt-lib/soft-eng.html`**
  Software Engineering Virtual Library in Houston

- **`http://www.mccabe.com/`**
  McCabe & Associates. Commercial site offering products and services for software developers (i. e. Y2K, Testing or Quality Assurance)

- **`http://www.sei.cmu.edu/`**
  Software Engineering Institute of the U. S. Department of Defence at Carnegie Mellon University. Main objective of the Institute is to identify and promote successful software development practices.
  Exhaustive list of publications available for download.

- **`http://dxsting.cern.ch/sting/sting.html`**
  Software Technology Interest Group at CERN: their WEB-service is currently limited (due to "various reconfigurations") to a list of links to other information sources.

- **`http://www.spr.com/index.htm`**
  Software Productivity Research, Capers Jones. A commercial site offering products and services mainly for software estimation and planning.

- **`http://www.qucis.queensu.ca/Software-Engineering/`**
  This site hosts the World-Wide Web archives for the USENET usegroup comp.software-eng. Some links to other information sources are also provided.

- **`http://www.esi.es/`**
  The European Software Institute, Spain

- **`http://www.lrgl.uqam.ca/`**
  Software Engineering Management Research Laboratory at the University of Quebec, Montreal. Site offers research reports for download. One key focus area is the analysis and extension of the Function Point method.

- **`http://www.SoftwareMetrics.com/`**
  Homepage of Longstreet Consulting. Offers products and services and some general information on Function Point Analysis.

- **`http://www.utexas.edu/coe/sqi/`**

Software Quality Institute of the University of Texas at Austin. Offers comprehensive general information sources on software quality issues.

- **http://wwwtrese.cs.utwente.nl/~vdberg/thesis.htm**
  Klaas van den Berg: Software Measurement and Functional Programming (PhD thesis)

- **http://divcom.otago.ac.nz:800/com/infosci/smrl/home.htm**
  The Software Metrics Research Laboratory at the University of Otago (New Zealand).

- **http://ivs.cs.uni-magdeburg.de/sw-eng/us/**
  Homepage of the Software Measurement Laboratory at the University of Magdeburg.

- **http://www.cs.tu-berlin.de/~zuse/**
  Homepage of Dr. Horst Zuse

- **http://dec.bournemouth.ac.uk/ESERG/bibliography.html**
  Annotaded bibliography on Object-Oriented Metrics

- **http://www.iso.ch/9000e/forum.html**
  The ISO 9000 Forum aims to facilitate communication between newcomers to Quality Management and those who have already made the journey have experience to draw on and advice to share.

- **http://www.qa-inc.com/**
  Quality America, Inc's Home Page offers tools and services for quality improvement. Some articles for download are available.

- **http://www.quality.org/qc/**
  Exhaustive set of online quality resources, not limited to software quality issues

- **http://freedom.larc.nasa.gov/spqr/spqr.html**
  Software Productivity, Quality, and Reliability N-Team

- **http://www.qsm.com/**
  Homepage of the Quantitative Software Management (QSM) in the Netherlands

- **http://www.iese.fhg.de/**
  Homepage of the Fraunhofer Institute for Experimental Software Engineering (IESE) in Kaiserslautern, Germany
- **http://www.highq.be/quality/besma.htm**
  Homepage of the Belgian Software Metrics Association (BeSMA) in Keebergen, Belgium

- **http://www.cetus-links.org/oo_metrics.html**
  Homepage of Manfred Schneider on Objects and Components

- **http://dec.bournemouth.ac.uk/ESERG/bibliography.html**
  An annotated bibliography of object-oriented metrics of the Empirical Software Engineering Research Group (ESERG) of the Bournemouth University, UK

News Groups

- **news:comp.software-eng**
- **news:comp.software.testing**
- **news:comp.software.measurement**

Software Measurement Associations

- **http://www.dasma.org**
  DASMA Deutsche Anwendergruppe für SW Metrik und Aufwandsschätzung e.V.

- **http://www.aemes.fi.upm.es**
  AEMES Association Espanola de Metricas del Software

- **http://www.cosmicon.com**
  COSMIC Common Software Measurement International Consortium

- **http://www.esi.es**
  ESI European Software Engineering Institute in Bilbao, Spain

- **http://www.mai-net.org/**
  Network (MAIN) Metrics Associations International

- **http://www.sttf.fi**
  FiSMA Finnish Software Metrics Association

- **http://www.iese.fhg.de**
  IESE Fraunhofer Einrichtung für Experimentelles Software Engineering
- **http://www.isbsg.org.au**
  ISBSG International Software Benchmarking Standards Group, Australia

- **http://www.nesma.nl**
  NESMA Netherlands Software Metrics Association

- **http://www.sei.cmu.edu/**
  SEI Software Engineering Institute Pittsburgh

- **http://www.spr.com/**
  SPR Software Productivity Research by Capers Jones

- **http://fdd.gsfc.nasa.gov/seltext.html**
  SEL Software Engineering Laboratory - NASA-Homepage

- **http://www.vrz.net/stev**
  STEV Vereinigung für Software-Qualitätsmanagement Österreichs

- **http://www.sqs.de**
  SQS Gesellschaft für Software-Qualitätssicherung, Germany

- **http://www.ti.kviv.be**
  TI/KVIV Belgish Genootschap voor Software Metrics

- **http://www.uksma.co.uk**
  UKSMA United Kingdom Software Metrics Association

Software Metrics Tools (Overviews and Vendors)

Tool Listings

- **http://www.cs.umd.edu/users/cml/resources/cmetrics/**
  C/C++ Metrics Tools by Christopher Lott

- **http://mdmetric.com/**
  Maryland Metrics Tools

- **http://cutter.com/itgroup/reports/function.html**
  Function Point Tools by Carol Dekkers

- **http://user.cs.tu-berlin.de/~fetcke/measurement/products.html**
  Tool overview by Thomas Fetcke

- **http://zing.ncsl.nist.gov/WebTools/tech.html**
  An Overview about Web Metrics Tools

Tool Vendors

- **http://www.mccabe.com**
  McCabe & Associates

- **`http://www.scitools.com`**
  Scientific Toolworks Inc.

- **`http://zing.ncsl.nist.gov/webmet/`**
  Web Metrics

- **`http://www.globalintegrity.com/csheets/metself.html`**
  Global Integrity

- **`http://www.spr.com/`**
  Software Productivity Research (SPR)

- **`http://jmetric.it.swin.edu.au/products/jmetric/`**
  JMetric

- **`http://www.imagix.com/products/metrics.html`**
  Imagix Power Software

- **`http://www.verilogusa.com/home.htm`**
  VERILOG (LOGISCOPE)

- **`http://www.qsm.com/`**
  QSM

METRICS NEWS

VOLUME 12 **2007** **NUMBER 1**

CONTENTS

ISSN 1431-8008