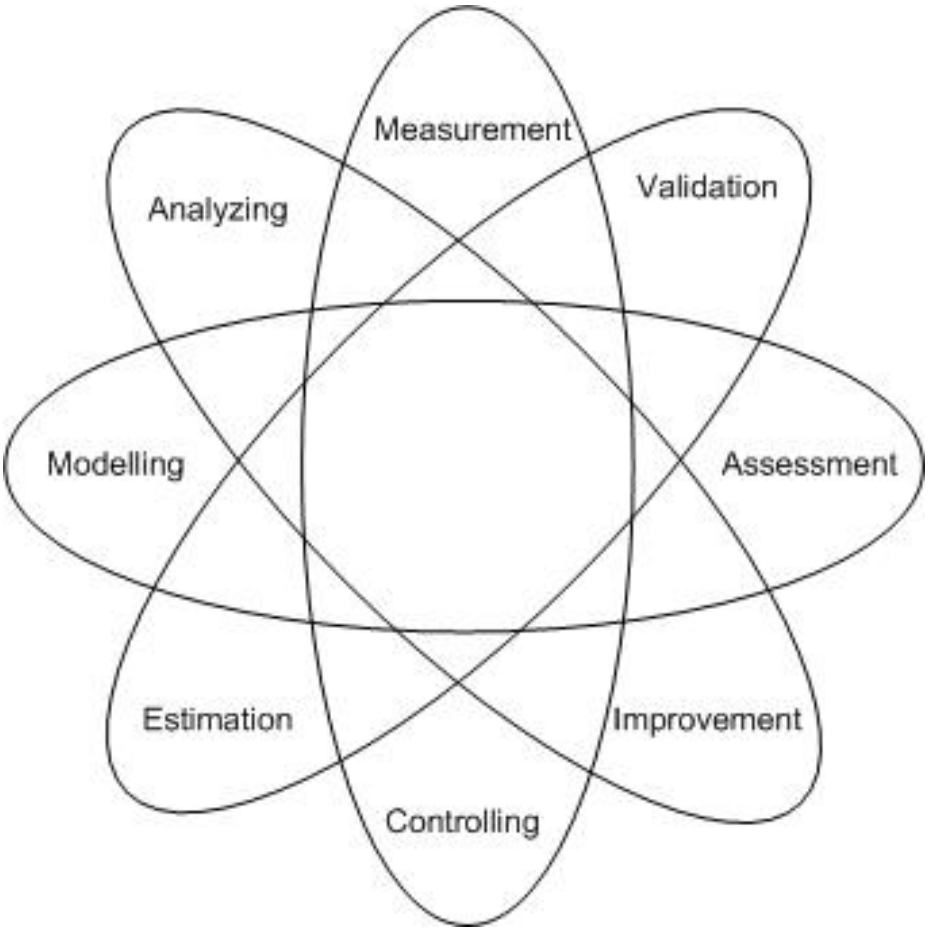
COSMIC

Volume 25, Number 2, September 2020

Software Measurement News

Journal of the Software Measurement Community



Editors:

Alain Abran, Manfred Seufert, Reiner Dumke, Christof Ebert, Cornelius Wille

SM L@b

Université du Québec
École de technologie supérieure

OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG

DASMA

CONTENTS

Announcements **2**
IWSM Mensura Conference 2020 2
ESAPI Workshop 2020 7
Announcements of the GUFPI-ISMA 11
Announcements of the NESMA 12
Announcements of the GI FG 2.1.10 13

Community Reports **14**

News Papers **17**

Charles Symons:
Why COSMIC functional Measurement? *17*
Caper Jones:
IFPUG Function Point Measurements *20*
Christof Ebert:
Test-Orientiertes Requirement Engineering *31*
Reiner Dumke, Anja Fiegler, Cornelius Wille:
COSMIC Examples – What could they mean as an IT project? 43

New Books on Software Measurement **53**

Conferences Addressing Measurement Issues **59**

Metrics in the World-Wide Web **62**

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, **alain.abran@etsmtl.ca**

Manfred Seufert
Chair of the DASMA, Median ABS Deutschland GmbH
Franz-Rennefeld-Weg 2, D-40472 Düsseldorf,
manfred.seufert@mediaan.com

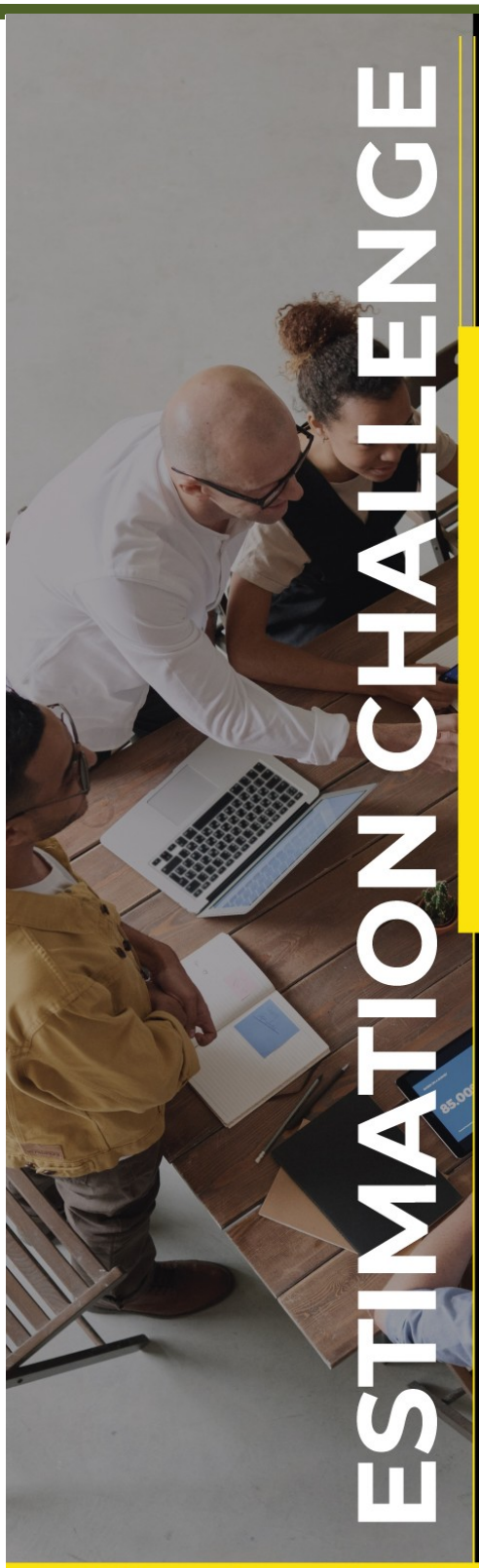
Reiner Dumke
Professor on Software Engineering, University of Magdeburg, FIN/IKS
Postfach 4120, D-39016 Magdeburg, Germany,
dumke@ivs.cs.uni-magdeburg.de, http://www.smlab.de

Christof Ebert
Dr.-Ing. in Computer Science, Vector Consulting Services GmbH
Ingersheimer Str. 20, D-70499 Stuttgart, Germany,
christof.ebert@vector.com

Cornelius Wille
Professor on Software Engineering, University of Applied Sciences Bingen
Berlinstr. 109, D-55411 Bingen am Rhein, Germany,
wille@fh-bingen.de

Editorial Office: University of Magdeburg, FIN, Postfach 4120, 39016 Magdeburg, Germany
Technical Editor: Dagmar Dörge
The journal is published in one volume per year consisting of two numbers. All rights reserved (including those of translation into foreign languages). No part of this issues may be reproduced in any form, by photo print, microfilm or any other means, nor transmitted or translated into a machine language, without written permission from the publisher.

© 2020 by Otto-von-Guericke-University of Magdeburg. Printed in Germany



Software Estimation Challenge - 2020

IWSM Mensura Conference, October 29 - 30, 2020

organized by COSMIC

Call for participation

Estimating software development effort is an essential part of getting new or improved software products to its intended end-users. An effort estimate sets the expectations of the end-users as to when the software will be ready and what it will cost. Multiple sources show that estimating software development effort has a bad track-record, despite over fourty years of research on good software estimating practices. The IWSM conference is devoted to promoting research and experience on good software estimating practices. To stimulate the use of good estimating practices, the IWSM conference is now challenging estimation practitioners and student teams to show their competence in estimating software development effort.

Estimating software projects is intellectually stimulating and challenging and worth winning an International Award!

The aim of this Challenge is to provide a rigorous interactive session in software effort estimation presentations by the participating teams on a realistic case study, with the added benefits of cross learning from teams and insights from professional experts panel.

Challenge is open to:

01 Practitioners:

(from private and public organizations)
-Case Study will be provided with a high-level set of software requirements, 4 weeks ahead of the challenge.
-Participants to present their Estimation Solution at the conference.

02 Students:

a) Case Study with detailed and constrained set of requirements will be provided, on the early morning of the Estimation Challenge Day.
b) Students will have to size and estimate through a regression analysis using the historical data provided relevant to the case study within a half-day time bound deadline.
c) Submit findings in a Powerpoint format to the Senior Review Panel.

COSMIC IWSM MENSURA + CNMES 2020

©Copyright COSMIC

Register your teams on the conference-challenge website at www.event2020.cnmes.mx no later than 30th september 2020. Forward questions to: Alain Abran [alain.abran@etsmtl.ca] or Onur Demirors [demirorso@gmail.com]

The pressure for more efficient software development and maintenance processes delivering appropriate quality is constantly increasing. Software measurement is a key technology with which to manage and to control software development projects and software maintenance activities.

Measurement is essential to any engineering activity, by increasing the scientific and technical knowledge for both the practice of software development and maintenance and for empirical research in software technology. For this reason the IWSM Mensura conference facilitates the exchange of software measurement experiences between academic theory and industry practice.

The International Workshop on Software Measurement (IWSM) has a big tradition that started in 1990 with a small working group in Germany with professor Reiner Dumke of the Otto von Güricke Universität in Magdeburg, Christof Ebert of Alcatel and professor Horst Zuse of the Technical University of Berlin.

During the nineties the team of professor Alain Abran from the University of Québec joined in. Until 2006 the IWSM rotated between the Montréal area in Canada and various German cities.

The International Conference on Software Process and Product Measurement (also known as Mensura, the roman word for to measure) was founded in 2006 by professor Juan J Cuadrado Gallego from the University of Alcalá (Spain). The first edition was held in Cádiz.

To foster research, practice and exchange of experiences and best practices in 2007 both conferences joined forces and held a joint conference on the Island of Mallorca. Since then both conferences act as a combined conference and the conference is now traveling around the world.

IWSM Mensura Conference Program

Thursday, October 29, 2020

9.00 to 10.30	***Estimation Challenge***
10.35 to 11.05	***Francisco Valdes Souto***: *Automation measurement functional software size with MENSURA*
11.05 to 11.15	***Estudies AMMS Panel***
12.40 to 13.10	J***orge Valeriano Assem:*** *Software project feasibility validation based on simulation of constraints*
13.10 to 13.40	***Dra. L izbeth Naranjo:*** *Confidence intervals in software estimates*
14.15 to 14.45	***Venus Padilla:*** *Project performance evaluation (time, cost and scope) using MENSURA*
14.45 to 15.15	***Estimation Challenge Awards***

Friday, October 30, 2020

(see the final plan)

Hassan Soubra, Yomma Abufrikha, Alain Abran: *Towards universal COSMIC size measurement automation*

Francisco Valdes-Souto, Jorge Valeriano-Assem: *Exploratory study: Simulating the productivity control in software projects using feedback loop control theory*

Wilder Perdomo-Charry, Julia Prior, John Leaney: *How to Columbian software companies evaluate software product quality?*

Francisco Valdes-Souto, Roberto Pedraza-Coello, F. C. Olguin-Barron: *COSMIC sizing of RPA software: A case study from a proof of concept implementation in a banking organization*

Nebi Yilmaz, Ayca Kolukisa Tarhan: *Meta-models for software quality and its evaluation: A systematic literature review*

Olga Ormandjieva, Mandana Omidbakhsh, Sylvie Trudel: *Measuring the 3V's of Big Data: A rigorous approach*

Philipp Haindl, Reinhold Plösch: *Specifying feature-dependent in an operational manner – Results from a case study with practitioners*

Hela Hakim, Asma Sellami, Hanene Ben Abdallah, Alain Abran: *Improving the structural size measurement method through the nested (multi-level) control structures assessment in UML sequence diagram*

Luigi Lavazza, Liu Geng, Roberto Meli: *Productivity of software enhancement projects: an empirical study*

Özden Özcan Top, Onur Demirors, Fergal Mc Caffery: *Challenges and working solutions in agile adaptation: Experiences from the industry*

Sylvie Trudel, Olga Ormandjieva: *Lean measurement: A proposed approach*

Harald van Heringen: *Portfolio cost estimation and performance measurement in the context of a SAFe scaled agile framework*

Sara Elmidaoui, Laila Chelkhi, Ali Idri, Alain Abran: *Predicting software maintainability using ensemble techniques and stacked generalization*

Abdelaziz Sahab, Sylvie Trudel: *COSMIC functional size automation of Java web applications using the Spring MVC framework*

Ahmed Darwish, Hassan Soubra: *COSMIC functional size of ARM assembly programs*

Duygu Deniz Erhan, Ayca Kolukisa Tarhan, Rana Özakinci: *Selecting suitable software effort estimation method*

ORGANIZADORES

MAIN SPONSORS

SUPPORT

MEDIA PARTNERS

VORLÄUFIGES PROGRAMM ZUM

WORKSHOP "EVALUATION OF SERVICE-APIS" – ESAPI 2020

APIS ALS „KLEBSTOFF" EINER ALLUMFASSENDEN DIGITALISIERUNG

03. NOVEMBER 2020 – BERLIN

ACHTUNG: Der Workshop wird virtuell über das Internet übertragen!

Gastgeber und Veranstaltungsort:

Konrad Nadobny

Bayer Aktiengesellschaft
R&D-Clinical Operations
Clinical Development Operations
13342 Berlin, Germany

Sprecher der ESAPI-Initiative:

Prof. Dr. Andreas Schmietendorf

Hochschule für Wirtschaft und Recht Berlin, FB II

Email: andreas.schmietendorf@hwr-berlin.de

Markus Bauer

Central Europe Computer Measurement Group

Email: markus.bauer@cecmg.de

Motivation:

Die Gartner Group[1] geht davon aus, dass im Jahr 2021 mehr als 60% aller Anwendungsentwicklungen von eingesetzten Web-APIs profitieren. Diese mit Hilfe klassischer Internettechnologien zur Verfügung gestellten Web-APIs bieten die Möglichkeit eines konsistenten Zugriffs auf fachlich begründete Informationen und Funktionen aber auch auf komplette Geschäftsprozesse. Neben einer unternehmens- und branchenübergreifenden Integration existierender Softwarelösungen wird dabei auch die Zielstellung einer kompositorischen und damit agilen Softwareentwicklung verfolgt. Aufgrund der ggf. „ad hoc“ zusammengesetzten Lösungen muss auch der Betrieb mit diesen Herausforderungen umgehen können. Daher kommt der Themenstellung „DevOps“ als Klammer zwischen Entwicklung und Betrieb eine besondere Bedeutung zu. Der diesjährige Workshop fokussiert die folgenden Themen:

[1] Bewertung von Vertrauen und Sicherheit bei Web-APIs.

[2] Branchenspezifische Ansätze zur Spezifikation von Web-APIs.

[3] Lowcode bzw. Codeless Softwareentwicklung mit Web-APIs.

[4] Effiziente Ansätze zur „API-fizierung“ von Altanwendungen.

[5] Risiken bei über Web-APIs bezogenen KI-Algorithmen.

[6] Vor- und Nachteile von GraphQL-basierten Web-APIs.

[7] Elemente eines DevOps-orientierten API-Managements

[8] Serverless bereitgestellte Web-APIs – Fiktion oder Wirklichkeit?

Weiteren Informationen und Anmeldung unter:

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

Aktuelle Informationen und Anmeldung:
http://www.cecmg.de oder sekretariat@cecmg.de

Bei Anmeldung bis zum 29. Oktober 2020 wird ein Unkostenbeitrag von 90,– € (ceCMG-, GI-, ASQF- und ASQT-Mitglieder: 70,– €) erhoben, danach 120,– € (ceCMG-, GI-, ASQF- und ASQT-Mitglieder: 100,– €). Über den Tagungsbeitrag erhalten Sie eine Rechnung der ceCMG e.V. (Central Europe Computer Measurement Group – Schirmherr der Veranstaltung). Studenten und Mitarbeiter des Gastgebers erhalten eine kostenfreie Teilnahme!

[1] Quelle: Zumerle, D. et al. 2019. API Security. What You Need to Do to Protect Your APIs [online]. Verfügbar unter https://www.gartner.com/en/documents/3956746/api-security-what-you-need-to-do-to-protect-your-apis

Agenda – Blended Workshop Concept (d.h. via Internet übertragen):

10:00 Uhr Eröffnung des Workshops

Prof. Dr. Andreas Schmietendorf (HWR Berlin & OvG-Universität Magdeburg)

Konrad Nadobny (Bayer AG, Digital Programming Manager)

Eröffnung des Workshops – Ziele und Motivation

10:15 Uhr Keynote 1

Anja Fiegler (Microsoft Deutschland GmbH)

Entwicklung smarter Anwendungen mit Hilfe cloudbasiert angebotener KI-Algorithmen

10:45 Uhr Pitches: Management & Architektur (jeweils 10 min.)

Dr. Niko Zenker, Marvin Leine (T-Systems)

Einsatz einer gewichteten Graphendatenbank zur Abbildung komplexer Unternehmensarchitekturen

Konrad Nadobny (Bayer AG)

Vergleich von Enterprise API-Management-Lösungen

Steven Schmidt (Deutsche Bahn AG)

Secure Public WIFI

11:15 Uhr Pause

11:45 Uhr Keynote 2

Michael Petry, Volker Reers, Dr. Frank Simon (Petry-Consulting, Reers-Consulting, Zurich Versicherungsgruppe Deutschland)

Reaktive, minimal destruktive API-Härtung am Beispiel Graph QL

12:15 Uhr Mittagspause

13:15 Uhr Keynote 3

Jens Borchers (Bfl Hamburg)

Zero Trust Infrastrukturen

13:45 Uhr Pitches: API-Security (jeweils 10 min.)

Daniel Kant, Prof. Dr. Andreas Johannsen (HS Brandenburg)

Exemplarische API-Schwachstellen bei IoT-Geräten auf Grundlage von OWASP API Security Top 10

Sandro Hartenstein, Gabriel Landa (HWR Berlin)

Bitcoin Blockchain via Satelliten Blockstream API

14:10 Uhr Pause

14:30 Uhr World-Cafés (Breakout Sessions) – Themenvorschläge

Massive APIfizierung von Legacy Applikationen
Herausforderungen beim KI-Bezug via Web-APIs
Vertrauen in Public WIFI-Infrastrukturen

15:30 Uhr Präsentation der Ergebnisse und Abschlussstatement

Bem.: Weitere Beiträge werden durch entsprechende Kurzvideos bzw. Poster auf der Internetpräsenz des Workshops vorgestellt. Änderungen der vorläufigen Agenda sind jederzeit möglich!

Medienpartner

Durch die folgenden Medienpartner wird der Workshop begleitet. Neben der Publizität geht es dabei auch um die begleitende Auslage der zum Workshop korrespondierenden Publikationen, aber auch um die Bereitstellung eines entsprechenden Tagungsbands.

SIGS DATACOM GmbH

Web: http://www.sigs-datacom.de

Shaker Verlag GmbH

Web: https://www.shaker.de

Programmkomitee

S. Aier, Universität St. Gallen
F. Balzer, Broadcom
M. Binzen, DB Systel GmbH
E. Dimitrov, T-Systems
R. Dumke, Uni Magdeburg
J. Marx Gómez, Uni Oldenburg
W. Greis, TPS Data & CECMG
J. Heidrich, Fraunhofer IESE
A. Johannsen TH Brandenburg
S. Kosterski, Toll Collect
M. Lother, Robert Bosch GmbH
P. Mandl, HS München
M. Mevius, HTWG Konstanz
H. Neumann, Deutsche Bahn AG
A. Nitze, Ultra Tendency UG
M. Rothaut, T-Systems Bonn
A. Schmietendorf, HWR Berlin
F. Simon, Zurich Insurance Group
F. Victor, TH Köln
C. Wille, TH Bingen
M. Weiß, HUK Coburg
T. Wiedemann, HTW Dresden
M. Wißotzki, HS Wismar
R. Zarnekow, TU Berlin

Announcement of the GUFPI-ISMA

GUFPI-ISMA *– the Italian Software Metrics Association (www.gufpi-isma.org) - is organizing next two virtual conferences (EventoMetrico) on September 18 and December 4 2020.*

The events (1-day long each), dealing with any kind of measurement related topic, will be set on the Zoom Webinar platform and will grant 5 PDUs (for those PMI-certified) and occasionally also IFPUG CECs for the CEP-program (for those CFPS-certified).

The conferences will be held in Italian and occasionally some presentation will be in English. Here the link for the September 18 event

(https://gufpiisma.wildapricot.org/widget/event-3802441)

with yet 260 people attending, including the program: would you like to join us?

Luigi Buglione

ISMA Presidente

Announcement of the NESMA

Inthe spotlight

•**Nesma Counting Practice is organizing a online meeting for experience function point analysts.** Please join if you want to experence more collaboration with your peers. The meeting is on Sept. 28, 2020 at 16.00 – 17.00.

•**Nesma would like to receive your feedback on a poll.** Please response with what should be the main focus of Nesma in 2021 – 2025. Click here to go to the poll in LinkedIN.

•**Eureka has resumed taking the CFPA examn.** This is possible by taking extra measures, such as keeping a distance. Click here to read more how they do that (in Dutch).

•**Applying the Nesma counting guidelines to web and internet development:** Remco van Beek of the Dutch Tax and Customs Administration (Belastingdienst) and his colleagues are developing an approach for the application of Nesma counting guidelines in Web and Internet development. They would like to share this approach with other function point analysts. In a knowledge-sharing session to be scheduled this spring, the participants will exchange their knowledge and experiences in this field. Click here for more information about the session and registration.

•**New version of the user guide for FPA in software enhancement projects:** recently Nesma has published an updated version of the user guide for the application of Function Point Analysis in software enhancement projects, both in English and in Dutch. The new versions can be downloaded free of charge from the Documents section of the website.

Announcement of the GI-FG 2.1.10

Unsere Fachgruppe widmet sich vor allem den Schwerpunkten zum Data Science aus Sicht des Big Data in den Ausprägungen:

- **als Big Data im Business-Bereich durch die enorm steigende Anzahl an Informationen und deren Verknüpfungsformen durch vielfältige Verflechtungen von Business-Prozessen in allen B2B, B2C und vor allem auch C2C Bereichen;**

- **durch die großen Mengen von (Software-) Messdaten als umfangreiche Repositories oder auch Qualitätssicherungsdaten für den Social Media Bereich und den zunehmend komplexeren IoT Systemen;**

- **in den Anwendungen der Big Data Analyseformen (als preskriptive und prediktive) auch in der Softwareanalytik sowie deren Nutzung als Deep Learning für relevante Entscheidungsprozesse**

Aktuelle Initiativen zu Foren, Workshops und Konferenzteilnahmen siehe unter

https://fg-metriken.gi.de

Currently COSMIC News

I am delighted to announce that the number of countries with a COSMIC representation has risen to 31 because of the addition of:

Cameroon
In Cameroon, COSMIC is represented by Donatien Moulla (donatien.moulla@cosmic-sizing.org). For more information, visit cosmic-sizing.org/organization/local/cameroon

Jordan
In Jordan, COSMIC is represented by Khalid Al-Sarayreh (khalid.alsarayreh@cosmic-sizing.org). For more information, visit cosmic-sizing.org/organization/local/jordan

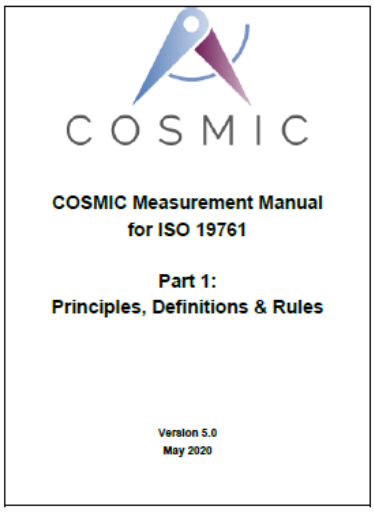
Morocco
In Morocco, COSMIC is now represented by Ali Idri (ali.idri@cosmic-sizing.org). For more information, visit cosmic-sizing.org/organization/local/morocco

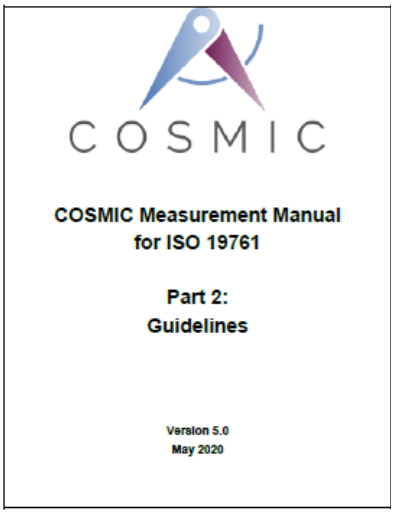
Frank Vogelezang
Chairman

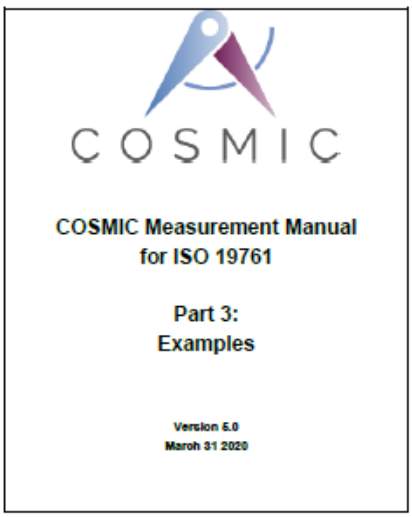
The COSMIC version 5.0 is now available

(see: www.cosmic-sizing.org/)

Part 1: *Principles, definitions & rules* (17 pages)*
Part 2: *Guidelines* (17 pages)*
Part 3: *Examples of COSMIC concepts and measurements (28 pages)*





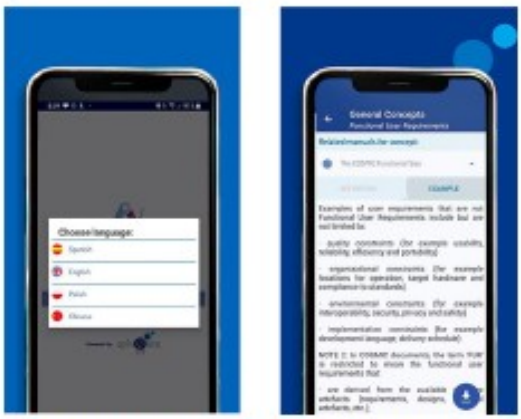


COSMIC DOCS

COSMIC DOSC APP

Powered by: spingere

The COSMIC DOSC APP (Supported by COSMIC) contains all the COSMIC sizing information for consulting and download, all the manuals, guidelines and concepts you need. Available in Playstore for sale, coming soon available on Appstore.



Four languages available and more coming soon:
English, Spanish, Chinese and Polish

All the COSMIC method information you need in a single app.

Download it!

Software Sizing Tools

There are a growing number of open source and commercial tools that have been created to support the COSMIC methodology. The COSMIC organisation takes no responsibility for these tools. All are web applications or SAAS unless otherwise stated.

Record/Measure CFP counts

- ScopeMaster *Sizer*
- SoftwareLite – App for Android
- Mensura
- VisualFSM – Windows Application

Automated COSMIC sizing from Requirements

- ScopeMaster – *Analyser, sizing and QA of user stories (English, Italian, Spanish & Portuguese).*
- ScopeMaster Story Analyser for Jira – App For Jira Cloud

Tools that perform early estimations in CFP

- Software Risk Master, Namcook Analytics – pre-requirements model-based estimation.
- ScopeMaster *Analyser* – estimations from automated requirements sizing.

Electronic Manual

- COSMIC Docs* – Mobile app version of the COSMIC Documentation. English and Spanish

Tools for CFP base estimation

- SEER for Software, Galorath – Windows Application
- SoftwareExpert – App for Android

Software Requirements Tools that can use CFP sizes

- Jira – custom fields, Trello – custom fields power up.

Why COSMIC Functional Measurement?

(cited from Symon's Measurement Guide by Reiner Dumke)

Charles Symons

"As a software professional, I want to learn to use the COSMIC method, the most powerful standard way to size software, so as to help improve our software processes, the quality of our software products, and our estimating"

"Knowing the size of your software is interesting, but *what you can do with measurements of software size is* ***valuable.***

If you know how to estimate or measure software sizes you have the basic means to:
- estimate the effort, time and then cost for a new development, early in its life;

- track **size** as the software is developed, to control 'scope creep' and manage risk;

- measure the productivity (= **size** / work-hours) and speed (= **size** / duration) with which the software was developed and is maintained;

- monitor the quality of the delivered software product (defect density = defects found in a given time-period / **size**)

- use measurements from across your software activities to learn how to improve organizational performance..... and more, limited only by your imagination!. "

"Why measure software size?

As in any other field of endeavour, size and scale matter. The larger the requirements for a new software system, the more expensive, risky, and difficult it will be to deliver. Similarly, the more an organization depends for its success on delivering software, the greater the challenges of managing project teams to deliver software on time, efficiently, and to acceptable quality.
So the ability to measure and understand the influence of size can be critical for the performance of a software-producing organization. Specifically, software size is a key parameter to measure and control the following tasks.

- ***Improving organizational performance in software development***. Knowing the size of some delivered software and the effort to develop it, you can calculate the productivity and speed of the development process.

(Productivity = size / effort; Speed = size / duration.)

And a count of defects discovered in a given time-period divided by size (i.e. 'defect density') is a valuable indicator of product quality.

The more performance data you gather, the more your organization can learn about the factors that influence performance favourably or unfavourably, which technologies are most productive, the possible trade-offs between effort, time and quality, etc., etc. That learning can be your foundation for improving performance.

- ***Estimating the cost of new developments***. If you can estimate an approximate size of the software early in its life and you know the productivity typically achieved by previous developments of similar software (otherwise known as 'benchmark' productivity), you can make a first estimate of the effort for the new development:

Estimated effort for new development = (Estimated size of new software) divided by (Productivity of previous developments).

Estimated development effort then becomes the main input to estimated development cost, cost/benefit analysis, budgeting, development planning, resource allocation, etc.

- ***Controlling software development***. If you can track the size of software under development as its requirements are worked out in more detail, then you have the means to control 'scope creep' and so help maintain the development cost within budget.

- ***Managing investments in software***. Evaluating the cost/benefit of a new investment, or deciding if and when it is economic to replace an existing system, etc., all require a good knowledge of system costs, and therefore of software sizes, the main cost-driver of software (re-)development.

A further benefit of measuring a COSMIC size early in the life of a new system is that ***the process leads to improved quality of the software requirements*** by helping identify ambiguities and inconsistencies, missing requirements, and suchlike. Users report that the insights gained from the measurement process lead to fewer product defects and hence lower development costs. Guide to Software Size Measurement V1.0, Copyright © 2020 9

Obviously the cost of gathering and using size and other data must be weighed against the potential value from pursuing the goals listed above. But for estimating and controlling medium/large software projects, there is no substitute for having 'hard' data for decision-making.

Remember the old adage '*you cannot manage what you cannot measure*'.

Different ways of measuring software size

The size of a piece of software can be measured in many ways. For example, you can:
- count the source lines of code (SLOC) of the software programs. However, SLOC counts are technology-dependent and are not much help for early cost estimation as you can only know the size accurately after the software exists;

- use methods such as Use Case Points, Object Points, etc., but these methods are not standardized and the resulting sizes depend on the software design;

- use Agile Story Points, but these are subjective. In practice, it's often not clear if they measure size, effort or duration, and their meaning varies with the individual agile team.

Above all, none of these methods can help yield all the possible benefits from measuring software sizes that we are aiming for.
If you do aim to gain these benefits, then the only choice is to measure a size of the required functionality of the software. A 'functional size' is based only on software requirements and so is totally independent of the technology, processes and the individuals or teams used to develop the software. 'Functional Size Measurement' (FSM) methods have been around for decades, but there is only one '2nd Generation' FSM method - the COSMIC method, the most powerful, generally-applicable, ISO-standard FSM method.

What uniquely distinguishes the COSMIC method?

The COSMIC method is the only functional size measurement method
- designed according to fundamental software engineering principles, and hence applicable to:

  + business, real-time and infrastructure software,
  + software in any layer of a software architecture, at any level of decomposition down to its smallest components,
  + in summary, any type of software where knowing its size is valuable;

- able to size requirements from single User Stories up to the requirements for whole releases or systems, with rules to ensure sizing consistency at all levels of aggregation;

- with a continuous, open-ended size scale that conforms with measurement theory.

- that is completely 'open' with all documentation available for free download.

A consequence of the method being based on fundamental software engineering principles is that its design is actually very simple. Essentially, if you can identify the underlying functional processes of the software and analyse them into four types of data movements (Entries, Exits, Reads and Writes), all as defined in this Guide, then you can measure COSMIC sizes."

IFPUG Function Point Based Measurements

(cited from Jones' "The Mess of Software Metrics" (May 2020) by Reiner Dumke, paper can be ordered from Capers.Jones3@gmail.com)

Capers Jones, Namcook Analytics LLC.

Keywords: *cost per defect, economic productivity, function points, lines of code (LOC), manufacturing economics, software productivity, SNAP metrics, software metrics, software quality, technical debt; function point metrics; function point 30th anniversary in 2017.*

Abstract

The software industry is one of the largest, wealthiest, and most important industries in the modern world. The software industry is also troubled by very poor quality and very high cost structures due to the expense of software development, maintenance, and endemic problems with poor quality control. Accurate measurements of software development and maintenance costs and accurate measurement of quality would be extremely valuable. But as of 2017 the software industry labors under a variety of non-standard and highly inaccurate measures compounded by very sloppy measurement practices. For that matter, there is little empirical data about the efficacy of software standards themselves. The industry also lacks effective basic definitions for "software productivity" and "software quality" and uses a variety of ambiguous definitions that are difficult to predict before software is released and difficult to measure after the software is released. This paper suggests definitions for both economic software productivity and software quality that are both predictable and measurable. The year 2017 marked the 30th anniversary of function point metrics. Function point metrics are the best available for measuring software economic productivity and software quality.

Introduction

The software industry has become one of the largest and most successful industries in history. However software applications are among the most expensive and error-prone manufactured objects in history. Software needs a careful analysis of economic factors and much better quality control than is normally accomplished. In order to achieve these goals, software also needs accurate and reliable metrics and good measurement practices. Unfortunately the software industry has ignored both.

The software industry has the worst metrics and measurement practices of any industry in human history. This is one of the reasons why the software industry has more failing projects than any other industry and a higher percentage large projects with cost and schedule overruns. It is also why a survey of CEO's in Fortune 500 companies think that software engineers are the least professional of any kind of engineers. Basically the software industry has been running blind for over 60 years due to harmful metrics such as "cost per defect" and "lines of code" both of which distort reality and conceal progress.

Fortunately function point metrics do measure both economic productivity and software quality. Calendar year 2017 marked the 30th anniversary of the International Function Point Users Group (IFPUG) which has become the largest metrics association in the industry. Other forms of function point metrics such as COSMIC, FISMA, NESMA, and automated function points from CAST software are also popular. Collectively function points are used for more software benchmarks than all other metrics combined. Table 1 shows comparative sizes in various metrics:

Table 1: Variations in Software Size Metrics 2020

(Based on 1,000 IFPUG 4.3 function points and Java language)
(Sizes predicted by Software Risk Master (SRM))

	Metrics	Nominal Size	SNAP Size	% of IFPUG Size
1	Automated code-based function points	1,070	142	107.00%
2	Automated UML-based function points	1,030	137	103.0%
3	Automated text-based function points	1,055	140	105.5%
4	Backfired function points	1,017	135	101.7%
5	Code size (logical statements)	53,000	7,049	NA
6	Code size (physical lines with comments, blanks)	145,750	19,385	NA
7	COSMIC function points	1,086	144	108.6%
8	Fast function points	970	129	97.0%
9	Feature points	1,000	133	100.0%
10	FISMA function points	1,020	136	102.0%
11	Full function points	1,170	156	117.0%
12	Function points light	967	129	96.7%
13	**IFPUG 4.3**	1,000	133	100.0%
14	IntegraNova function points	1,090	145	109.0%
15	Mark II function points	1,060	141	106.0%
16	NESMA function points	1,040	138	104.0%
17	Object-Oriented function points (OOFP)	735	98	73.5%
18	RICE objects	4,439	590	443.9%
19	SCCQI function points	2,877	383	287.7%
20	Simple function points	975	130	97.5%
21	SNAP non functional size metrics	133		13.3%
22	**SRM pattern matching function points**	1,000	133	100.0%
23	Story points	333	44	33.3%
24	Unadjusted function points	890	118	89.0%
25	Use-Case points	200	27	20.0%
26	Weighted micro function points	1,127	134	112.7%

But more work is needed because in 2020 over half of software development companies and over 70% of government software organizations still use invalid metrics such as LOC and cost per defect.

Fortunately a number of countries are starting to mandate function points for government contracts: Brazil, Malaysia, Italy, South Korea, and Japan. Others will probably do the same in future years.

This paper deals with some of the most glaring problems of software metrics and suggests a metrics and measurement suite that can actually explore software economics and software quality with high precision. The suggested metrics can be predicted prior to development and then measured after release. The key metrics in this suite include 1) function points, 2) work hours per function point, 3) defect potentials using function points, 4) defect removal efficiency (DRE), and also three quality metrics 5) delivered defects per function point; 6) high-severity defects per function point; and 7) security flaws per function point.

Supplemental metrics include pre-relase and post-release application growth using function points and dollar costs per function point for development, maintenance, cost of quality (COQ), and total cost of ownership (TCO. Application size grows at about 1% per month during development and about 8% per year after release. SNAP metrics for non-functional requirements are also discussed but have very little data even today, although it is growing.

Following are descriptions of the more common software metric topics in alphabetical order:

Backfiring is a term that refers to mathematical conversion between lines of code and function points. This method was first developed by A.J. Albrecht and colleagues during the original creation of function point metrics, since the IBM team had LOC data for the projects they used for function points. IBM used logical code statements for backfiring rather than physical LOC. There are no ISO standards for backfiring. Backfiring is highly ambiguous and varies by over 500% from language to language and company to company. A sample of "backfiring" is the ratio of about 106.7 statements in the procedure and data divisions of COBOL for one IFPUG function point. Consulting companies sell tables of backfire ratios for over 1000 languages, but the tables are not the same from vendor to vendor. Backfiring is not endorsed by any of the function point associations. Yet probably as many as 100,000 software projects have used backfiring because it is quick and inexpensive, even though very inaccurate with huge variances from language to language and programmer to programmer.

Benchmarks in a software context often refer to the effort and costs for developing an application. Benchmarks are expressed in a variety of metrics such as "work hours per function point," "function points per month," "lines of code per month," "work hours per KLOC," "story points per month," and many more. Benchmarks also vary in scope and range from project values, phase values, activity values, and task values. There are no ISO standards for benchmark contents. Worse, many benchmarks "leak" and omit over 50% of true software effort. The popular benchmark of "design, code, and unit test" termed DCUT contains only about 30% of total software effort. The most common omissions from benchmarks include unpaid overtime, management, and the work of part-time specialists such as technical writers and software quality assurance. Thus benchmarks from various sources such as ISBSG, QSM, and others cannot be directly compared since they do not contain the

same information. The best and most reliable benchmarks feature activity-based costs and include the full set of development tasks; i.e. requirements, architecture, business analysis, design, coding, testing, quality assurance, documentation, project management, etc.

Cost estimating for software projects is generally inaccurate and usually optimistic. About 85% of projects circa 2017 use inaccurate manual estimates. The other 15% use the more accurate parametric estimating tools of which these are the most common estimating tools in 2015, shown in alphabetical order: COCOMO, COCOMO clones, CostXpert, ExcelerPlan, KnowledgePlan, SEER, SLIM, Software Risk Master (SRM), and TruePrice. A study by the author that compared 50 manual estimates against 50 parametric estimates found that only 4 of the 50 manual estimates were within plus or minus 5% and the average was 34% optimistic for costs and 27% optimistic for schedules. For manual estimates, the larger the projects the more optimistic the results. By contrast 32 of the 50 parametric estimates were within plus or minus 5% and the deviations for the others averaged about 12% higher for costs and 6% longer for schedules. Conservatism is the "fail safe" mode for estimates. The author's SRM tool has a patent-pending early sizing feature based on pattern matching that allows it to be used 30 to 180 days earlier than the other parametric estimation tools. It also predicts topics not included in the others such as litigation risks, costs of breach of contract litigation for the plaintiff and defendant, and document sizes and costs for 20 key document types such as requirements, design, user manuals, plans, and others. The patent-pending early sizing feature of SRM produces size in a total of 23 metrics including function points, story points, use case points, logical code statements, physical lines of code, and many others.

Cost per defect metrics penalize quality and makes the buggiest software look cheapest. There are no ISO or other standards for calculating cost per defect. Cost per defect does not measure the economic value of software quality. The urban legend that it costs 100 times as much to fix post-release defects as early defects is not true and is based on ignoring fixed costs. Due to fixed costs of writing and running test cases, cost per defect rises steadily because fewer and fewer defects are found. This is caused by a standard rule of manufacturing economics: "*If a manufacturing process has a high percentage of fixed costs and there is a reduction in the units produced, the cost per unit will go up*." This explains why cost per defect seems to go up over time even though actual defect repair costs are flat and do not change very much. There are of course very troubling defects that are expensive and time consuming, but these are comparatively rare. Appendix A explains the problems of cost per defect metrics.

Defect removal efficiency (DRE) was developed by IBM circa 1970. The original IBM version of DRE measured internal defects found by developers and compared them to external defects found by clients in the first 90 days following release. If developers found 90 bugs and clients reported 10 bugs, DRE is 90%. This measure has been in continuous use by hundreds of companies since about 1975. However there are no ISO standards for DRE. The International Software Benchmark Standards Group (ISBSG) unilaterally changed the post-release interval to 30 days in

spite of the fact that the literature on DRE since the 1970's was based on a 90 day time span, such as the author's 1991 version of Applied Software Measurement and his more recent book on The Economics of Software Quality with Olivier Bonsignour. Those with experience in defects and quality tracking can state with certainty that a 30 day time window is too short; major applications sometimes need more than 30 days of preliminary installation and training before they are actually used. Of course bugs will be found long after 90 days; but experience indicates that a 90-day interval is sufficient to judge the quality of software applications. A 30 day interval is not sufficient.

Earned value management (EVM) is a method of combining schedule, progress, and scope. It originated in the 1960's for government contracts and has since been applied to software with reasonable success. Although earned value is relatively successful, it really needs some extensions to be a good fit for software projects. The most urgent extension would be to link progress to quality and defect removal. Finding and fixing bugs is the most expensive software activity. It would be easy to include defect predictions and defect removal progress into the earned value concept. Another extension for software would be to include the specific documents that are needed for large software applications. If the earned-value approach included quality topics, it would be very useful for contracts and software outsource agreements. EVM is in use for defense software contracts, but the omission of quality is a serious problem since finding and fixing bugs is the most expensive single cost driver for software. The U.S. government requires earned value for many contracts. The governments of Brazil and South Korea require function points for software contracts. Most projects that end up in court for breach of contract do so because of poor quality. It is obvious that combining earned-value metrics, defect and quality metrics, and function point metrics would be a natural fit to all software contracts and would probably lead to fewer failures and better overall performance.

Defect density metrics measure the number of bugs released to clients. There are no ISO or other standards for calculating defect density. One method counts only code defects released. A more complete method used by the author includes bugs originating in requirements, architecture, design, and documents as well as code defects. The author's method also includes "bad fixes" or bugs in defect repairs themselves. There is more than a 500% variation between counting only released code bugs and counting bugs from all sources. For example requirements defects comprise about 20% of released software problem reports.

Function point metrics were invented by IBM circa 1975 and placed in the public domain circa 1978. Function point metrics do measure economic productivity using both "*work hours per function point*" and "*function points per month*". They also are useful for normalizing quality data such as "defects per function point". However there are numerous function point variations and they all produce different results: Automatic, backfired, COSMIC, Fast, FISMA, IFPUG, Mark II, NESMA, Unadjusted, etc. There are ISO standards for COSMIC, FISMA, IFPUG, and NESMA. However in spite of ISO standards all four produce different counts. Adherents of each function point variant claim "accuracy" as a virtue but there is no cesium atom or independent

way to ascertain accuracy so these claims are false. For example COSMIC function points produce higher counts than IFPUG function points for many applications but that does not indicate "accuracy" since there is no objective way to know accuracy.

Goal/Question metrics (GQM) were invented by Dr. Victor Basili of the University of Maryland. The concept is appealing. The idea is to specify some kind of tangible goal or target, and then think of questions that must be answered to achieve the goal. This is a good concept for all science and engineering and not just software. However, since every company and project tends to specify unique goals the GQM method does not lend itself to either parametric estimation tools or to benchmark data collection. It would not be difficult to meld GQM with function point metrics and other effective software metrics such as defect removal efficiency (DRE). For example several useful goals might be "*How can we achieve defect potentials of less than 1.0 per function point?*" or "*How can we achieve productivity rates of 100 function points per month?*" Another good goal which should actually be a target for every company and every software project in the world would be "*How can we achieve more than 99% in defect removal efficiency (DRE)?*"

ISO/IEC standards are numerous and cover every industry; not just software. However these standards are issued without any proof of efficacy. After release some standards have proven to be useful, some are not so useful, and a few are being criticized so severely that some software consultants and managers are urging a recall such as the proposed ISO/IEC testing standard. ISO stands for the International Organization for Standards (in French) and IEC stands for International Electrical Commission. While ISO/IEC standards are the best known, there are other standards groups such as the Object Management Group (OMG) which recently published a standard on automatic function points. Here too there is no proof of efficacy prior to release. There are also national standards such as ANSI or the American National Standards Institute, and also military standards by the U.S. Department of Defense (DoD) and by similar organizations elsewhere. The entire topic of standards is in urgent need of due diligence and of empirical data that demonstrates the value of specific standards after issuance. In total there are probably several hundred standards groups in the world with a combined issuance of over 1000 standards, of which probably 50 apply to aspects of software. Of these only a few have solid empirical data that demonstrates value and efficacy.

Lines of code (LOC) metrics penalize high-level languages and make low-level languages look better than they are. LOC metrics also make requirements and design invisible. There are no ISO or other standards for counting LOC metrics. About half of the papers and journal articles use physical LOC and half use logical LOC. The difference between counts of physical and logical LOC can top 500%. The overall variability of LOC metrics has reached an astounding 2,200% as measured by Joe Schofield, the former president of IFPUG! LOC metrics make requirements and design invisible and also ignore requirements and design defects, which outnumber code defects. Although there are benchmarks based on LOC, the intrinsic errors of LOC metrics make them unreliable. Due to lack of standards for counting LOC,

benchmarks from different vendors for the same applications can contain widely different results. Appendix B provides a mathematical proof that LOC metrics do not measure economic productivity by showing 79 programming languages with function points and LOC in a side-by-side format.

SNAP point metrics are a new variation on function points introduced by IFPUG in 2012. The term SNAP is an awkward acronym for "software non-functional assessment process." The basic idea is that software requirements have two flavors: 1) functional requirements needed by users; 2) non-functional requirements due to laws, mandates, or physical factors such as storage limits or performance criteria. The SNAP committee view is that these non-functional requirements should be sized, estimated, and measured separately from function point metrics. Thus SNAP and function point metrics are not additive, although they could have been. Having two separate metrics for economic studies is awkward at best and inconsistent with other industries. For that matter it seems inconsistent with standard economic analysis in every industry. Almost every industry has a single normalizing metric such as "cost per square foot" for home construction or "cost per gallon" for gasoline and diesel oil. As of 2016 none of the parametric estimation tools have fully integrated SNAP and it may be that they won't since the costs of adding SNAP are painfully expensive. As a rule of thumb non-functional requirements are about equal to 15% of functional requirements, although the range is very wide.

Story point metrics are widely used for agile projects with "user stories." Story points have no ISO standard for counting or any other standard. They are highly ambiguous and vary by as much as 400% from company to company and project to project. There are few useful benchmarks using story points. Obviously story points can't be used for projects that don't utilize user stories so they are worthless for comparisons against other design methods. (The author's Software Risk Master (SRM) estimating tool converts story points to function points and agile sprints into a standard chart of accounts. These conversions allow agile to compared side by side against DevOps, Iterative, Container development, waterfall, spiral, etc.)

Taxonomy of software applications is needed to ensure "apples-to-apples" benchmark comparisons. Although there are several taxonomies for software, the one developed by the author is useful for sizing, estimating, and benchmark data collection. It is a standard feature in the author's Software Risk Master (SRM) tool. The elements of the SRM taxonomy include: 1) Country code, 2) Region code, 3) City code, 4) Industry code (we use the North American Industry Classification code or NAIC code), 5) Project nature; 6) Project scope, 7) Project class, 8) Project type, 9)Project hardware platform, 10) Problem complexity, 11) Code complexity, 12) Data complexity. It happens that projects with identical taxonomies are usually very similar, which makes benchmark comparisons interesting and useful. We also include some additional topics of interest: A) Methodology chosen from a list of 50; B) Programming languages chosen from a list of 180; C) Project CMMI level; D) Team experience of several kinds; E) Project management experience; F) Client experience; G) Reusable materials available.

Technical debt is a new metric and rapidly spreading. It is a brilliant metaphor developed by Ward Cunningham. The concept of "technical debt" is that topics deferred during development in the interest of schedule speed will cost more after release than they would have cost initially. However there are no ISO standards for technical debt and the concept is highly ambiguous. It can vary by over 500% from company to company and project to project. Worse, technical debt does not include all of the costs associated with poor quality and development short cuts. Technical debt omits canceled projects, consequential damages or harm to users, and the costs of litigation for poor quality.

Use case points are used by projects with designs based on "use cases" which often utilize IBM's Rational Unified Process (RUP). There are no ISO standards for use cases. Use cases are ambiguous and vary by over 200% from company to company and project to project. Obviously use cases are worthless for measuring projects that don't utilize use cases, so they have very little benchmark data. This is yet another attempt to imitate the virtues of function point metrics, only with somewhat less rigor and with imperfect counting rules as of 2015.

Velocity is an agile metric that is used for prediction of sprint and project outcomes. It uses historical data on completion of past work units combined with the assumption that future work units will be about the same. Of course it is necessary to know future work units for the method to operate. The concept of velocity is basically similar to the concept of using historical benchmarks for estimating future results. However as of 2015 velocity has no ISO standards and no certification. There are no standard work units and these can be story points or other metrics such as function points or use case points, or even synthetic concepts such as "days per task." If agile projects use function points then they could gain access to large volumes of historical data using activity-based costs; i.e. requirements effort, design effort, code effort, test effort, integration effort, documentation effort, etc. Story points have too wide a range of variability from company to company and project to project; function points are much more consistent across various kinds of projects. Of course COSMIC, IFPUG, and the other variants don't have exactly the same results.

Further interesting tables:

Table 2: Software Quality for 1000 Function Points, Java, and Agile Development

Defect Potentials	Number of Bugs	Defects Per FP
Requirements defects	750	0.75
Architecture defects	150	0.15
Design defects	1,000	1.00
Code defects	1,350	1.35
Document defects	250	0.25
Sub Total	**3,500**	**3.50**
Bad fixes	150	**0.15**
TOTAL	**3,650**	**3.65**
Defect removal Efficiency (DRE)	**97.00%**	**97.00%**
Defects removed	3,540	3.54
Defects delivered	**110**	**0.11**
High-severity delivered	**15**	**0.02**

Table 4: Average Software Defect Potentials circa 2020 for the United States

- Requirements 0.70 defects per function point
- Architecture 0.10 defects per function point
- Design 0.95 defects per function point
- Code 1.15 defects per function point
- Security code flaws 0.25 defects per function point
- Documents 0.45 defects per function point
- Bad fixes 0.65 defects per function point
- **Totals 4.25 defects per function point**

References and Readings
Books and monographs by Capers Jones.

1 Jones, Capers; The Technical and Social History of Software Engineering; Addison Wesley 2014
2 Jones, Capers & Bonsignour, Olivier; The Economics of Software Quality; Addison Wesley, 2012
3 Jones, Capers; Software Engineering Best Practices; 1st edition; McGraw Hill 2010
4 Jones, Capers: Applied Software Measurement; 3rd edition; McGraw Hill 2008
5 Jones, Capers: Estimating Software Costs, 2nd edition; McGraw Hill 2007
6 Jones, Capers: Software Assessments, Benchmarks, and Best Practices; Addison Wesley, 2000
7 Jones, Capers: Software Quality - Analysis and Guidelines for Success, International Thomson 1997
8 Jones, Capers; Patterns of Software Systems Failure and Success; International Thomson 1995
9 Jones, Capers; Assessment and Control of Software Risks; Prentice Hall 1993
10 Jones, Capers: Critical Problems in Software Measurement; IS Mgt Group 1993

Monographs by Capers Jones 2012-2017 available from Namcook Analytics LLC

1 Comparing Software Development Methodologies
2 Corporate Software Risk Reduction
3 Defenses Against Breach of Contract Litigation
4 Dynamic Visualization of Software Development
5 Evaluation of Common Software Metrics
6 Function Points as a Universal Software Metric
7 Hazards of "cost per defect" metrics
8 Hazards of "lines of code" metrics
9 Hazards of "technical debt" metrics
10 History of Software Estimation Tools
11 How Software Engineers Learn New Skills
12 Software Benchmark Technologies
13 Software Defect Origins and Removal Methods
14 Software Defect Removal Efficiency (DRE)
15 Software Project Management Tools

Software Benchmark Providers (listed in alphabetic order)

1	4SUM Partners	www.4sumpartners.com
2	Bureau of Labor Statistics, Department of Commerce	www.bls.gov
3	Capers Jones (Namcook Analytics LLC)	www.namcook.com
4	CAST Software	www.castsoftware.com
5	Congressional Cyber Security Caucus	cybercaucus.langevin.house.gov
6	Construx	www.construx.com
7	COSMIC function points	www.cosmicon.com
8	Cyber Security and Information Systems	https://s2cpat.thecsiac.com/s2cpat/
9	David Consulting Group	www.davidconsultinggroup.com
10	Forrester Research	www.forrester.com
11	Galorath Incorporated	www.galorath.com
12	Gartner Group	www.gartner.com
13	German Computer Society	http://metrics.cs.uni-magdeburg.de/
14	Hoovers Guides to Business	www.hoovers.com
15	IDC	www.IDC.com
16	ISBSG Limited	www.isbsg.org
17	ITMPI	www.itmpi.org
18	Jerry Luftman (Stevens Institute)	http://howe.stevens.edu/index.php?id=14
19	Level 4 Ventures	www.level4ventures.com
20	Namcook Analytics LLC	www.namcook.com
21	Price Systems	www.pricesystems.com
22	Process Fusion	www.process-fusion.net
23	QuantiMetrics	www.quantimetrics.net
24	Quantitative Software Management (QSM)	www.qsm.com
25	Q/P Management Group	www.qpmg.com
26	RBCS, Inc.	www.rbcs-us.com
27	Reifer Consultants LLC	www.reifer.com
28	Howard Rubin	www.rubinworldwide.com
29	SANS Institute	www.sabs,org
30	Software Benchmarking Organization (SBO)	www.sw-benchmark.org
31	Software Engineering Institute (SEI)	www.sei.cmu.edu
32	Software Improvement Group (SIG)	www.sig,eu
33	Software Productivity Research	www.SPR.com
34	Standish Group	www.standishgroup.com
35	Strassmann, Paul	www.strassmann.com
36	System Verification Associates LLC	http://sysverif.com
37	Test Maturity Model Integrated	www.experimentus.com

Test-Orientiertes Requirement Engineering

Christof Ebert, Vector

(see also: C. Ebert "Systematisches Requirements Engineering", 2019)

Abstract:
Im Projekt gibt es zwei Wege zum Friedhof: Perfektion und Schlampigkeit. Agiles Testen mit testbaren Anforderungen, die direkt als Basis für das Testmanagement dienen, vereinfacht die Softwareentwicklung. Vorteil: Die Anforderungen sind verständlich, testbar, und direkt als Testfall anwendbar. Durchlaufzeit und Kosten im Test werden um bis zu dreißig Prozent reduziert. Dieser Artikel zeigt pragmatisch und mit vielen Beispielen, wie das geht.

1 Requirements und Test

Requirements Engineering und Test gehören zusammen. Historisch haben Tester die Anforderungen oft erst zu Gesicht bekommen, wenn das System bereits teilweise realisiert war. Das hatte zwei gravierende Nachteile. Zum einen kam unzureichende Anforderungsqualität viel zu spät auf den Tisch. Zum anderen war es ein ziemlicher Mehraufwand, ohne den Kontext der Anforderungsermittlung passende Testfälle abzuleiten. Viel Zusatzaufwand und lange Korrekturschleifen waren die Folge.

Test hat die unangenehme Eigenschaft, dass er immer zu teuer ist, und in der Regel massiv Aufwand verschwendet. In vielen Projekten wird für Verifikation und Validierung über die Hälfte der Lebenszyluskosten bezahlt. Mehr Test ist nicht mehr wert. Testfälle zu sammeln und ständig zu wiederholen bringt nichts, wenn dahinter keine gute Teststrategie steht. Unsere Projekte von Vector Consulting mit ganz verschiedenen Kunden zeigen, dass Test in der Regel künstlich und losgelöst von den kritischen Anwendungsszenarien ist [Ebe19]. Das Falsche wird getestet, und Abdeckungslücken werden spät oder gar nicht erkannt.

Nur eine agile Balance von geplanter Abdeckung und früher Testvorbereitung führt zum Erfolg [You18, You20]. Solch risikoorientiertes Arbeiten optimiert auch das Requirements Engineering. Statt Paralyse durch Analyse wird das spezifiziert, was nötig ist. Mein Kollege und Freund Al Davis nannte das „Just enough requirements engineering" [Dav05, You20].

Gute Tester brauchen negative Ergebnisse. Dies liegt nicht daran, dass sie negativ eingestellt sind, sondern daran, dass sie für den Misserfolg bezahlt werden. Ein gelungener Testfall zeigt einen Fehler. Ein Testfall, der erfolgreich läuft, zeigt ohne Teststrategie rein gar nichts. Viele Testfälle zu sammeln wie weiland Briefmarken, mag zwar manch unbegabten Entwicklern ein Gefühl der Sicherheit geben, aber Menge erhöht zunächst nur die Kosten – ohne Aussicht auf Erfolg. Schlimmer noch, Tester wählen unter Stress in Regressions-Tests die falschen Kombinationen aus. Klasse statt Masse ist auch im Test die Devise. Die Teststrategie muss daher ständig hinterfragt werden und risikoorientiert optimiert werden [Cha18, Hus16, Ebe14, Bja16].

Testorientiertes Requirements Engineering (***TORE***) verbessert die Anforderungsqualität und macht damit das Projekt von Beginn an schlanker und effizienter. Der Grund dafür ist einfach. Ein Tester fragt grundsätzlich bei jeder Anforderung und in jedem Szenario zuerst einmal: „Was wäre, wenn ...?“ Falls die Antwort klar und nachvollziehbar ist, wird er weiter fragen, bis eine Situation auftritt, die noch nicht ausreichend spezifiziert oder gar analysiert und abgestimmt ist. Erst dann kann er zufrieden sein und wird die gefundenen Schwachpunkte in seiner Liste der Auffälligkeiten markieren.

Die parallele Entwicklung von Anforderungen und Testfällen verbessert die Anforderungen bereits in deren Entstehungsprozess. Das reduziert Fehler in den Anforderungen und verringert drastisch den Abstimmungsaufwand im Projekt aufgrund missverstandener Anforderungen. Das sichert Testbarkeit der Anforderungen sowie eine frühzeitige Betrachtung der Umsetzbarkeit und des nötigen Integrationsaufwands. Durchgängiges Testen, gezielte Testabdeckung und effiziente Regressionstests sind damit möglich.

Testorientiertes Requirements Engineering bildet verschiedene Arten von Anforderungen, wie Funktionen, Qualitätsanforderungen und Randbedingungen frühzeitig in der Analyse auf passende Testfälle ab. Nur diese klare Aufteilung verschiedener Arten von Anforderungen und Testfällen stellt sicher, dass Fehler frühestmöglich in der Phase gefunden werden, in der sie zuerst auftreten. Damit ist TORE auch eine Erweiterung des risikobasierten Testens, denn aus hochpriorisierten Anforderungen oder auch solchen mit kritischem Bezug, wie Safety und Security, werden unmittelbar passende Testfälle abgeleitet.

Dieser Beitrag beschreibt die praktische Umsetzung von testorientiertem Requirements Engineering (TORE) in der Praxis. Testfälle werden parallel zu den Anforderungen entwickelt, und damit die Machbarkeit der Anforderungen viel schneller analysiert als in der traditionellen sequenziellen Vorgehensweise, in der Tests relativ spät spezifiziert werden. Die Testfälle werden dabei zunächst in der gleichen Struktur wie die Anforderungen als Ergänzung zu Anforderungen beschrieben. Damit verlagert sich das als agile Methodik bereits bewährte Test-Driven Development (TDD) auf die Spezifikationsebene. Regressionstests werden attributiert, um die spätere Automatisierung vorbereiten zu können. Der für den Test nötige Aufwand ist auf dieser Basis besser schätzbar, und damit sind die Projekt- und Qualitätsrisiken reduziert. Dieser Beitrag basiert auf Methodik und einem Beispiel, das im Buch „Systematisches Requirements Engineering“ beschrieben wurde [Ebe19].

2 Testorientierung in der Praxis

Gutes Requirements Engineering braucht gute Tester, die im Analyse- und Spezifikationsprozess aktiv beteiligt sind. Umgekehrt gilt, dass ein brauchbarer Test mit hoher Effektivität und Produktivität bereits in der Spezifikationsphase beginnt. Effizient wird das Requirements Engineering, wenn die Anforderungen nicht nur testbar sind, sondern auch direkt als Testfälle nutzbar sind [Ebe19, Bja16].

Die Anforderungsermittlung beeinflusst den späteren Test. Abb. 1 zeigt die grundlegenden Zusammenhänge im testorientierten RE. Die zentrale Frage hier lautet: Was ist der wesentliche Nutzen für verschiedene Anspruchsträger? Danach richtet sich nicht nur die Priorisierung und Iterationsplanung, sondern auch die Testplanung. Schließlich müssen die gelieferten Funktionen im fertigen System mit der richtigen Qualität zur Verfügung stehen. Jede darunterliegende Schicht geht zwar mehr ins Detail, weitet dabei aber immer das RE und den damit verbundenen Test aus. Die Systemanalyse liefert ein Pflichtenheft oder Fachkonzept bei IT-Lösungen, das als Basis für den Systemtest aus einer internen Sicht dient. Der Systementwurf legt die Hardware- und Softwareanforderungen fest. Entsprechend werden diese Funktionen später im Integrationstest verifiziert.

Testorientiertes RE gewährleistet, dass für jede Abstraktionsebene (horizontale Schichten in Abb. 1) sowohl die Anforderungen als auch die wesentlichen Testfälle spezifiziert werden. Beim TORE wird insbesondere die obere Ebene betrachtet, um Systemanforderungen direkt mit Integrations- und Systemtests zu verknüpfen. Nur diese klare Aufteilung verschiedener Arten von Anforderungen und Testfällen stellt sicher, dass Fehler frühestmöglich in der Phase gefunden werden, in der sie zuerst auftreten. Damit steht auch das Gerüst der Regressionstest für spätere Änderungen und agiles Arbeiten fest [You18, You20].

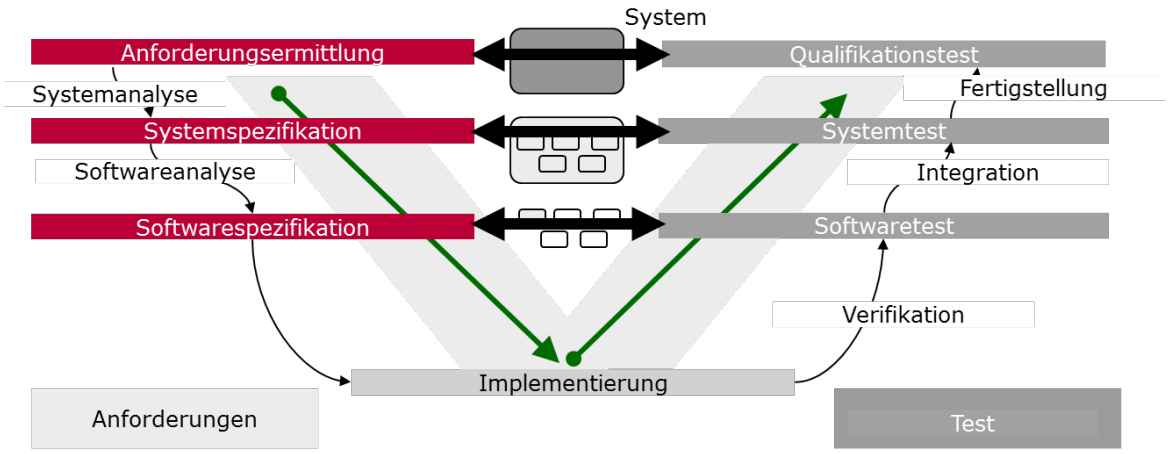


Abb. 1 Testorientiertes Requirements Engineering

Wie wird testorientiertes Requirements Engineering (TORE) umgesetzt? Vereinfacht gesagt wird jede Anforderung bereits in der Ermittlung als Testfall spezifiziert. Das kann in zwei Methoden erfolgen:

1. Man schreibt die Anforderung unmittelbar als Testfall, ähnlich wie man im Test-Driven Development (TDD) zuerst den Unit Testfall schreibt, und dann den Code, der diesen Test erfüllt.

2. Man nimmt die klassische Kundenanforderung wie sie ist, und schreibt dazu einen oder mehrere Testfälle, die diese Anforderung unmittelbar testbar machen und die wesentliche Funktionalität abdecken. Korrelationen von Anforderungen können offensichtlich in dieser Phase nicht beachtet werden.

Sofern die Anforderungen formalisiert sind, können die Testfälle automatisch aus strukturierten Anforderungen generiert werden. Das ist bei Anforderungen mit konsistenter Struktur, wie Use Cases mit messbaren Vor- und Nachbedingungen, mit Skripts möglich, wie wir es beispielsweise in IT-Systemen oder auch beim Test von GUI durchführen [Ebe19].

Wir wollen dieses Vorgehen und seinen Nutzen an einem kleinen Beispiel aus der Gebäudeautomatisierung veranschaulichen. iHome ist unser Beispiel einer intelligenten und vernetzten Automatisierung, die wir komplett durchgängig realisiert haben. Nehmen wir die folgende Anforderung aus iHome:

> *M-Req-1: Bei der Wartung sind die Konfigurationsdaten von iHome automatisch so zu präsentieren wie beim letzten Mal.*

Die Motivation für diese Anforderung ist Benutzbarkeit. Eine mühsam optimierte Einstellung für einen bestimmten Bearbeiter sollte nicht jedes Mal manuell neu eingestellt werden müssen. Nun kommt der Tester ins Spiel. Er entdeckt sofort einige offene Punkte in der Spezifikation: Was ist das „letzte Mal“? Sollen die Einstellungen pro Szenario oder pro Benutzer eingefroren werden? Sind die Sichten beliebig und autonom durch den Benutzer definierbar oder werden sie aus vorkonfigurierten Sichten durch Parametrisierung individualisiert? Sind alle Sichten für alle Benutzer verfügbar oder aber nur diejenigen, die ein Benutzer vorher für sich selbst definiert hatte? Er wird viele solcher Subtilitäten finden, die später im Projekt bestenfalls zu Klärungsbedarf führen und damit das Projekt verzögern, und schlimmstenfalls zu viel Verwirrung und Fehlern, wenn sie ohne Klärung nach Gutdünken der Entwickler realisiert werden. Nachdem diese offenen Punkte geklärt sind, wird der Tester interne Akzeptanzkriterien vorschlagen und daraus Testfälle ableiten.

Ein zugehöriger Testfall könnte wie folgt aussehen:

> ***Testfall-1**: Ändere die Darstellung eines Admin-Bildschirms von Zoom, Fensteranordnung und gewählten Ausschnitten. Verlasse die Szenario-Beschreibung. Gehe zur Beschreibung zurück. Die Beschreibung wird exakt gleich dargestellt.*

Mit diesem Testfall wird die Anforderung klar, und man kann sie trotzdem so formulieren, dass sie nicht überbestimmt ist.

Der klare und konsistente Bezug zwischen Anforderungen und Testfällen unterstützt das Testdatenmanagement. Abb. 2 zeigt exemplarisch einen solchen Zusammenhang, wie er in vielen Produkten auftritt. Links ist die Produktsicht dargestellt, zu der eine bestimmte (versions- oder variantenabhängige) Testliste und projektbegleitend die jeweiligen Testergebnisse (z.B. Reports, Testabdeckung, Fehlerzahlen, Korrekturfortschritt, Regressionstests) gehören. Rechts ist – projektübergreifend – die Testbibliothek dargestellt, die es erlaubt, Testfälle in verschiedenen Varianten und Versionen einzusetzen. Konsistenz wird durch die Verweise auf bestimmte Produktanforderungen erreicht.

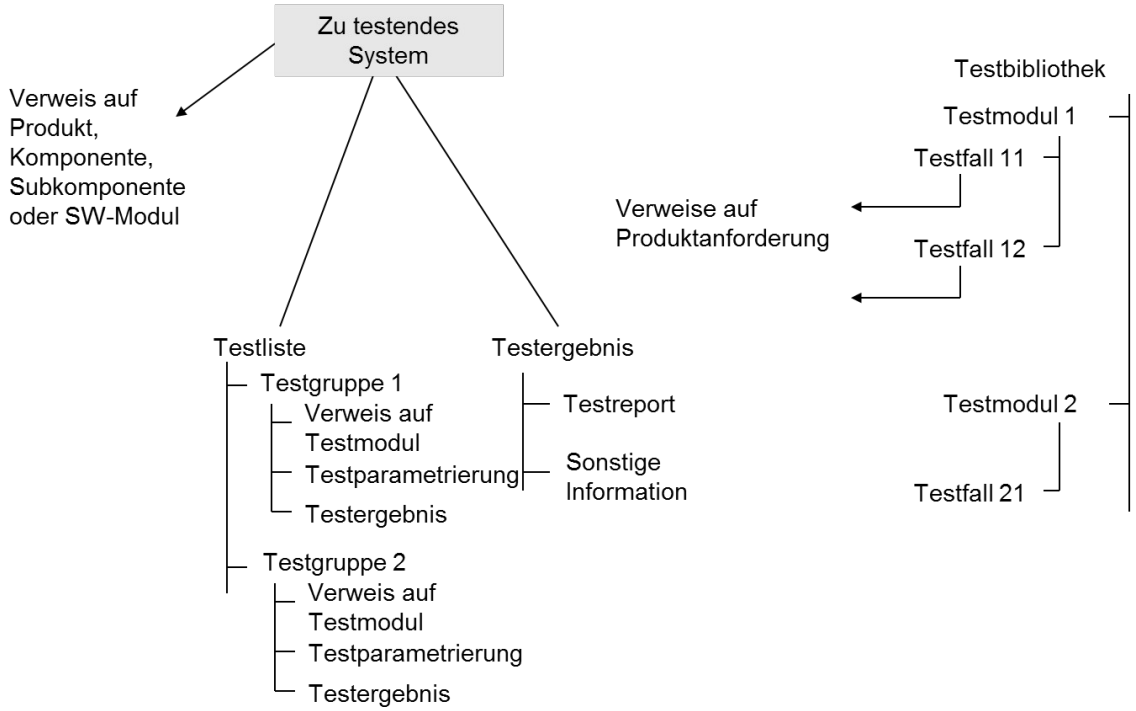


Abb. 2 Testdatenmanagement mit Produktanforderungen als gemeinsamer Referenz

Testorientiertes RE sollte direkt im ALM oder PLM durch eine frühzeitige Verknüpfung von Anforderungen mit Testfällen unterstützt werden. Ohne adäquate Werkzeugunterstützung werden die Inhalte schnell inkonsistent, da in der Hektik des Alltags einfach die Zeit fehlt, mehrere Quellen parallel zu pflegen. Mit Kunden bauen wir oft spezifische Umgebungen für deren gewachsene Werkzeugumgebung auf, um TORE effizient umzusetzen [Ebe19]. Abb. 3 zeigt ein solches Beispiel aus einer konkreten Werkzeugumgebung. Wir nutzen hier eine industrielle ALM Umgebung, die RE, Modellierung und Test unterstützt. Über eine Traceability werden die Anforderungen unmittelbar als Testfälle für den Vorwärtsfall spezifiziert, und dann im Rahmen der Teststrategie mit Korrelationen ergänzt. Regressionstests werden markiert, um spätere Änderungen effizient und dennoch zielgenau abzusichern.

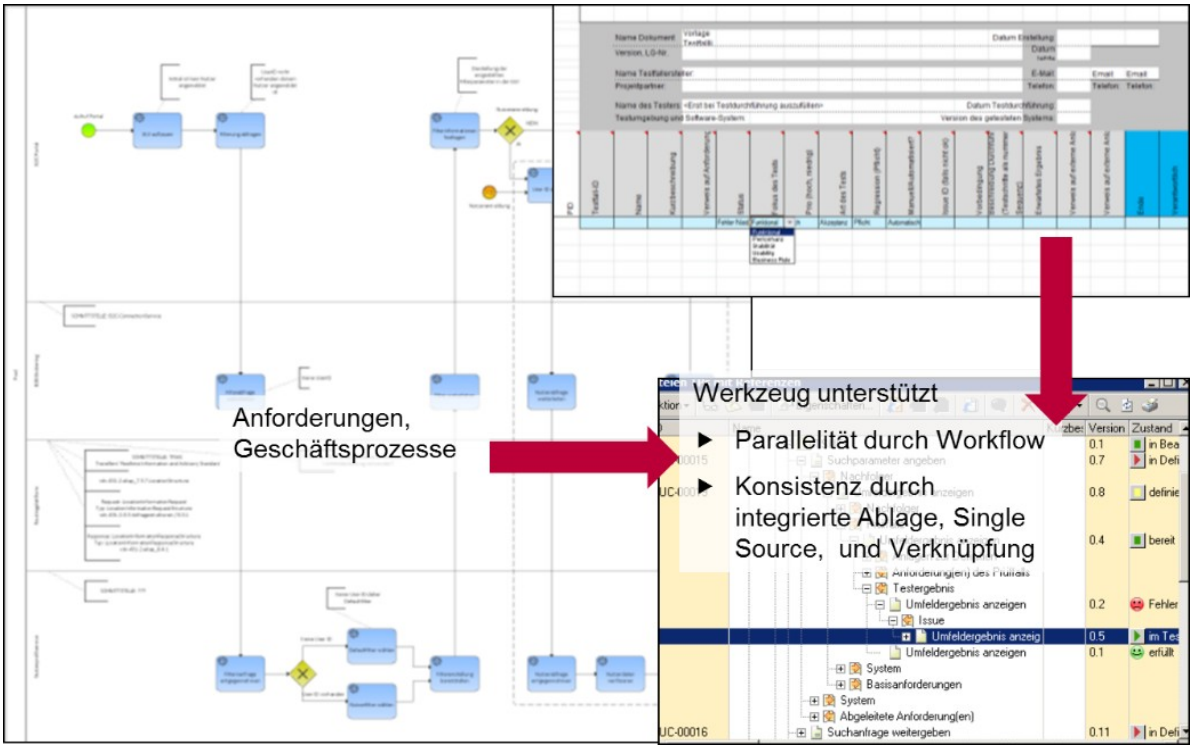


Abb. 3 Werkzeugunterstützung in ALM/PLM für Testorientiertes Requirements Engineering

3 Nutzen von TORE

Testorientiertes RE hat einen hohen Kosten-Nutzen Effekt, da es die Aufwände in Analyse und Prüfung frühzeitig auf das Testen lenkt und damit Doppelarbeit reduziert. Das Planen und Spezifizieren von ersten konkreten Testfällen helfen offensichtlich beim Spezifizieren und Verstehen von Anforderungen.

Vollständigkeit der Anforderungen.

Tester bemerken sofort, wenn es nicht definierte Bereiche gibt. Sie fragen – und prüfen sehr pragmatisch: Wie soll ich das denn testen? Was wird hier erwartet? Und was passiert, wenn der Benutzer „xyz“ eingibt? Was werden Nichtfachleute an dieser Stelle eingeben? Woher erhält dieses System das File „abc“?

Genauigkeit und Klarheit der Anforderungen.

Oftmals bleiben Anforderungen oberflächlich, weil sie einen faulen Kompromiss enthalten oder weil die Details noch nicht vollständig abgestimmt sind. Damit kann man kein System entwerfen. Zumindest müssen Aspekte beschrieben werden, die noch offen sind, damit man eventuell eine Parametrisierung vorsehen kann. Tester bemerken solche Oberflächlichkeiten und verlangen nach mehr Genauigkeit.
Betrachten wir beispielhaft Prüfpunkte, die einem Tester sofort auffallen, aber selten einem Entwickler:

- Erwartet das Szenario eine Eingabe, die unter gewissen Randbedingungen in einem anderen Szenario noch gar nicht abgeschlossen ist?
- Wird hier ein Signal zur Synchronisation eingesetzt, welches nicht immer stabil zur Verfügung steht?
- Kann sich die Währung, Zeit oder allgemein ein Datenformat aufgrund von Lieferantenwechsel ändern?
- Was passiert, wenn an dieser Stelle der Vorgang abgebrochen wird?
- Wie verhält sich das System, wenn während der Transaktion der Server nicht reagiert?
- Kann das CPU-Board während einer Transaktion ausgetauscht werden?

Prüfbarkeit der Anforderungen.

Anforderungen können genau und vollständig sein und dennoch nicht testbar, weil bestimmte Voraussetzungen noch nicht quantitativ präzisiert sind. Tester sind vor allem daran interessiert, in den Anforderungen auch die späteren Abnahmekriterien wiederzufinden. Nur diese Präzision erlaubt es ihnen, die Testfälle zu optimieren und nicht mit zu vielen Testfällen das Produkt unnötig teuer zu machen. Typische Fragestellungen sind: Wie soll denn nun die Wartbarkeit nachgewiesen werden? Brauchen wir eine bestimmte Testinfrastruktur, um die geforderte Effizienz zu erreichen? Wie wird der Kunde die gewünschte Gebrauchstauglichkeit konkret feststellen? Gibt es bestimmte Anwendungsfälle, die für den Kunden besonders wichtig sind? Welche Ausnahmesituationen müssen getestet werden? An welchen Szenarien verdient der Kunde später am meisten Geld?

Überspezifizierung und Randbedingungen abschwächen.

Die Beteiligung von Testern in der Spezifikationsphase verhindert zu starke Randbedingungen. Oftmals werden alle nicht gewünschten Szenarien per Definition ausgeblendet. Dies mag zwar aus Benutzer- und aus Spezifikationssicht einfach sein, reduziert aber auch den verfügbaren Lösungsraum. Tester hinterfragen Randbedingungen und Qualitätsanforderungen genauso wie funktionale Anforderungen. Schließlich bringt jede Randbedingung ebenfalls eine Menge von Testfällen mit sich.

4 Warum ist der Tester wichtig?

Tester werden dafür bezahlt, Fehler zu finden. In der Spezifikationsphase ist diese Einstellung sehr viel wert, denn die meisten anderen Beteiligten wollen nur die Phase schnellstmöglich abschließen – wohlwissend, dass sich Unstimmigkeiten nachher im Projekt zeigen werden. TORE nutzt die limitierten Kapazitäten der Tester bestmöglich für besser Anforderungsqualität und später schnelleren und wirksameren Test.

Ein Tester fragt grundsätzlich bei jeder Anforderung und in jedem Szenario zuerst einmal: „Was wäre, wenn ...?“ Falls die Antwort klar und nachvollziehbar ist, wird er weiter fragen, bis eine Situation auftritt, die noch nicht ausreichend spezifiziert oder gar analysiert und abgestimmt ist. Erst dann kann er zufrieden sein und wird die gefundenen Schwachpunkte in seiner Liste der Auffälligkeiten markieren.

Dieses Vorgehen zeigt, dass gute Tester an negativen Ergebnissen interessiert sind. Dies liegt nicht daran, dass sie negativ eingestellt sind, sondern daran, dass sie für den Misserfolg bezahlt werden.

Negative Anforderungen benötigen eine passende Prüfung. Die Validierung mit Testfällen genügt nicht. Dazu müssen mit Techniken wie der Gefahrenanalyse, der Fehlerbaumanalyse oder der FMEA Szenarien konstruiert und analysiert werden, die nicht eintreten dürfen. Daraus lassen sich dann Testfälle und Abnahmekriterien entwickeln. Allerdings zeigen diese Testfälle nicht, ob das System wirklich sicher oder fehlerfrei ist. Nur spezifische Verifikationstechniken können dies prüfen, beispielsweise ein Erreichbarkeitsgraph, der nachweist, dass ein bestimmter ausgeschlossener Zustand unter keinen Bedingungen erreicht wird.

Schauen wir nochmals auf ein Beispiel aus der oben eingeführten Hausautomatisierung iHome. Wir haben die folgende Anforderung:

> ***M-Req-2:*** *Bei geöffneten Fenstern lässt sich die Alarmanlage nicht aktivieren.*

Daraus wird die Produktanforderung abgeleitet:

> ***P-Req-21:*** *Wenn der Fenstersensor den Zustand „offen“ zeigt, ist Alarmanlage nicht aktivierbar.*

Im testorientierten RE wird nun nach einem passenden Testfall gesucht, der das ursprüngliche Ziel aus der Marktanforderung abdeckt. Geprüft wird diese Konstellation durch einen abgeleiteten Testfall (ist die Alarmlage bei geöffneten Fenstern inaktiv?) und eine Prüfung (gibt es einen Zustand, in dem die Alarmanlage aktivierbar ist, obwohl ein Fenster offen ist?). Offensichtlich gibt es solche Zustände, beispielsweise bei defekten Sensoren. Diese einfache Prüfung zeigt, dass P-Req-21 unvollständig war.

In sicherheitskritischen Systemen sowie in Systemen, in denen kritische Zustände nicht eintreten dürfen, ist eine risikoorientierte durchgängige Prüfung aus Haftungsgründen zwingend vorgeschrieben. Dazu wird eine Gefahrenanalyse durchgeführt, aus der Sicherheitsziele abgeleitet werden. Diese Schutzziele sind initiale Systemanforderungen, die weiter in Komponentenanforderungen untergliedert werden. Eine „brute force“ Validierung durch noch so viele Testfälle genügt nicht, solange keine methodische Abdeckung dieser Schutzziele nachvollziehbar ist. Abb. 4 zeigt einen Ausschnitt der Gefahrenanalyse von iHome. Und die methodische Ableitung der Schutzziele, die dann in Testfälle umgewandelt werden.

System: iHome Personenaufzug
Subsystem: Seilzugfeststellbremse

1) Kritischer Use Case: Was muss unbedingt funktionieren?
   Beispiel: Aus der Funktionsbeschreibung von Servo-Mechanik, Seilzug, Elektromotor und Ansteuerelektronik lassen sich kritische Use-Cases oder mögliche Unfallszenarien identifizieren. Nach Abwärtsfahrt mit vielen Haltepunkten steht der Aufzug und wird durch die Seilzugfeststellbremse abgesichert. Die aufgebrachte Spannkraft sichert die Kabine gegen minimale Bewegungen.
2) Fehlfunktionen: Welche Fehlfunktionen können auftreten? Methoden: HAZARD-Analysis, System-FMEA und Komponenten-FMEA, Misuse Cases
   Beispiel: Die Spannkraft kann sich physikalisch beim Abkühlen des Seilbremssystems so weit verringern, dass die Kabine nicht mehr ausreichend gesichert ist.
3) Bewertung: Wie kritisch ist die jeweilige Fehlfunktion? Methoden: FTA, FMEDA, Zuverlässigkeitsdiagramme.
   Beispiel: Aufzüge können Personenschäden verursachen. Daher Bewertung mit SIL 3.
4) Schutzziel: Wie muss eine noch fehlende Produktanforderung aussehen?
   Beispiel: Nach jeder Bewegung muss die Kabine ausreichend gesichert sein.
5) Umsetzung: Wie muss die zugehörige Komponentenanforderung aussehen?
   Beispiel: Aus den Schutzzielen leiten sich mit den Wie-Fragen die Sicherheitsanforderungen auf Komponentenebene ab. „Welche Sicherheitsfunktion kann die Fehlfunktion kompensieren?“ „Wie lässt sich die Sicherheitsfunktion umsetzen?“
   Beispiel: Die beteiligten Elektronikkomponenten überwachen Kabinenbewegungen auch nach Stillstand, um Seilverkürzungen durch Abkühlung entgegenzuwirken.

Abb. 4 Beispiel: Gefahrenanalyse

Systemtests und Qualifikationstests müssen bereits zu Projektbeginn konstruiert werden. Nur dann darf man davon ausgehen, dass die Anforderungen aus Kundensicht die richtige Qualität haben. Daher ist es für den Projekterfolg relevant, dass Tester bereits in der Analysephase zur Verfügung stehen. Das Problem dabei ist, dass die gleichen Tester in der Anforderungsphase benötigt werden und dort gerade in der heißen Phase des Vorgängerprojekts Überstunden aufhäufen. Abhilfe schaffen dabei dezidierte Expertenteams, die langfristig gebildet werden. In diesen Expertenteams sind verschiedene Expertisen fest eingeplant, die mit einem bestimmten Prozentsatz für die Analyse und für Reviews der Spezifikationen in den frühen Phasen von Projekten zur Verfügung stehen. Beispielsweise optimiert ein Tester die Teststrategie durch Eliminieren von Redundanzen, kritikalitätsbasiertes unbalanciertes Testen, etc..

Die Testabdeckung verknüpft Anforderungen mit der Umsetzung und zeigt, wie viele Anforderungen bereits getestet sind und wie gut sie funktionieren (Abb. 5). Damit steuert sie sowohl den Tester, der versucht, sie möglichst hoch zu trimmen, als auch den Auftraggeber, der sicherstellen will, dass sein Produkt adäquat getestet wurde.

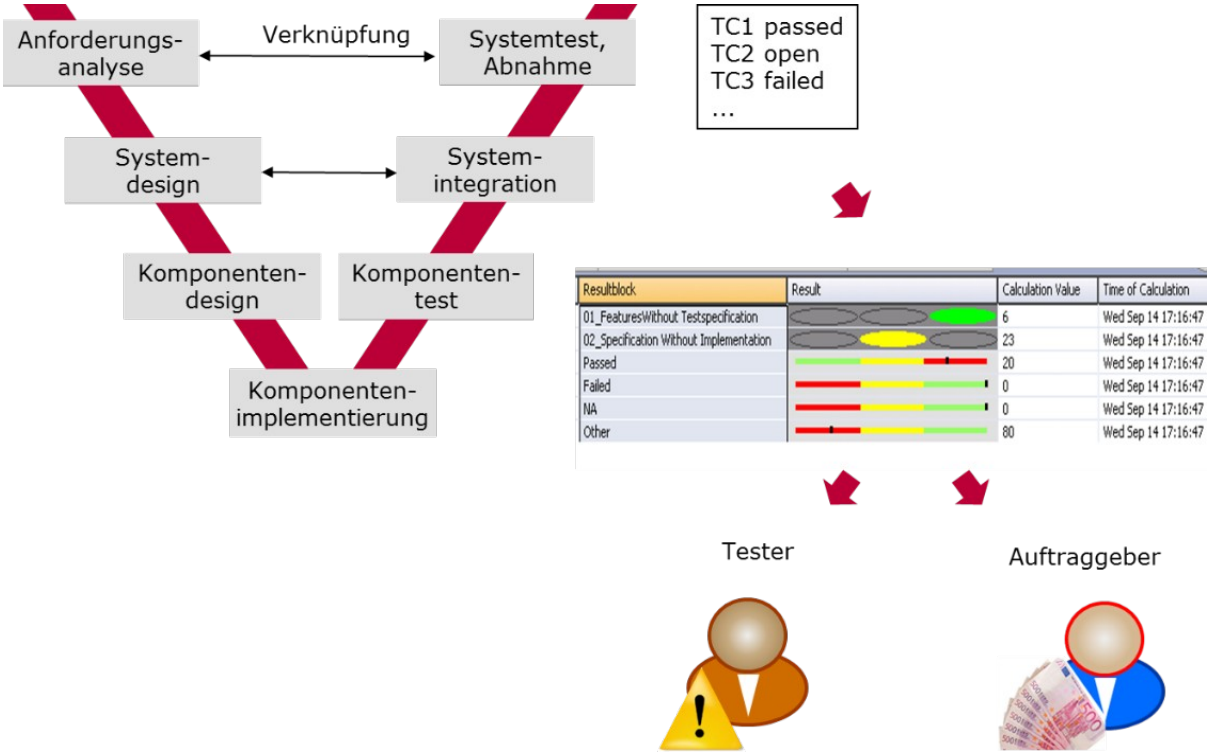


Abb. 5 Testabdeckung und effiziente Projektsteuerung

5 Zusammenfassung

TORE ist eine Methode, die agile Software-Entwicklung nach „oben“ hin zum Test erweitert. Wo bisher vor allem Unit Testing als Spezifikation von Code genutzt wurde, abstrahiert TORE hin zu Integrationstests und Systemtests im Zusammenspiel mit Anforderungen derselben Abstraktionseben.

Zum Abschluss einige Praxis-Tipps aus unseren Projekten:

- Jede einzelne Funktionsanforderung hat mindestens ein Abnahmekriterium, das entweder erfüllt oder nicht erfüllt ist.
- Jede einzelne Qualitätsanforderung wird mit numerischen Werten beschrieben, die gemessen werden können.
- Geschäftsregeln werden so definiert, dass festgestellt werden kann, ob sie wahr oder falsch sind.
- Geschäfts- und Datenobjekte werden mit all ihren Attributen, Typen und Zuständen definiert, so dass diese zur Testzeit gesetzt und validiert werden können.

- Systemschnittstellen, z.B. GUIs, Berichte und Service-Schnittstellen, werden in das Anforderungsdokument aufgenommen, so dass ihnen Werte zugewiesen werden können.

- Alle Anwendungsfälle haben Vor- und Nachbedingungen, die generiert und validiert werden können.

- Der gesamte Text wird markiert, so dass er automatisch zur Generierung von Testfällen verarbeitet werden kann.

In unseren Projekten mit Kunden verschiedener Branchen, aber auch in Vector für die eigene Software-Entwicklung, hat TORE verschiedene greifbare Nutzen gezeigt [Ebe19]:

- Vollständigkeit der Anforderungen. Tester bemerken sofort, wenn es nicht definierte Bereiche gibt. Sie fragen und prüfen sehr pragmatisch: Wie soll ich das denn testen? Was wird hier erwartet? Und was passiert, wenn der Benutzer „xyz“ eingibt? Was werden Nichtfachleute an dieser Stelle eingeben? Woher erhält dieses System das File „abc“?

- Genauigkeit und Klarheit der Anforderungen. Oftmals bleiben Anforderungen oberflächlich, weil sie einen faulen Kompromiss enthalten oder weil die Details noch nicht vollständig abgestimmt sind. Damit kann man kein System entwerfen. Zumindest müssen Aspekte beschrieben werden, die noch offen sind, damit man eventuell eine Parametrisierung vorsehen kann. Tester bemerken solche Oberflächlichkeiten und verlangen nach mehr Genauigkeit.

- Prüfbarkeit der Anforderungen. Anforderungen können genau und vollständig sein und dennoch nicht testbar, weil bestimmte Voraussetzungen noch nicht quantitativ präzisiert sind. Tester sind vor allem daran interessiert, in den Anforderungen auch die späteren Abnahmekriterien wiederzufinden. Nur diese Präzision erlaubt es ihnen, die Testfälle zu optimieren und nicht mit zu vielen Testfällen das Produkt unnötig teuer zu machen.

- Randbedingungen abschwächen. Die Beteiligung von Testern in der Spezifikationsphase verhindert zu starke Randbedingungen. Oftmals werden alle nicht gewünschten Szenarien per Definition ausgeblendet. Dies mag zwar aus Benutzer- und aus Spezifikationssicht einfach sein, reduziert aber auch den verfügbaren Lösungsraum. Tester hinterfragen Randbedingungen und Qualitätsanforderungen genauso wie funktionale Anforderungen. Schließlich bringt jede Randbedingung ebenfalls eine Menge von Testfällen mit sich.

Dieser Beitrag hat eine Übersicht zur Methodik und Stand der Technik des testorientierten Requirements Engineering dargestellt. Gleichzeitig zeigen unsere Industrieerfahrungen aus verschiedenen Branchen von der IT bis zu sicherheitskritischer eingebetteter Software, wie die Methodik wirksam umgesetzt wird. Die beschriebenen Lessons Learned aus der Praxis helfen beim Transfer in die eigene Umgebung.

6 Referenzen

[Bja16] Bjarnason, E. et al., *"A Multi-case Study of Agile Requirements Engineering and the Use of Test Cases as Requirements,"* Information and Software Technology, Sept. 2016, pp. 61–79.

[Cha18] Charette, R. N.: *Puncturing Pernicious Project Pufferies*. IEEE Computer, Vol. 51, No. 5, pp. 78-83, 2018.

[Dav05] Davis, A. M.: *Just Enough Requirements Management.* Dorset House, New York, USA, 2005.

[Ebe14] Ebert, C.: *Risikomanagement.* Springer, 2. Auflage, 2014.

[Ebe19] Ebert, C.: *Requirements Engineering*. dPunkt Verlag, 6. Auflage, 2018.

[Hus16] Hussain, A., E.Mkpojiogu, F.M.Kamal: *The Role of Requirements in the Success or Failure of Software Projects. International Review of Management and Marketing,* No. S7, Vol. 6, pp. 306-311, 2016.

[You18] *Agile Requirements Engineering.* YouTube video: https://youtu.be/svTb2X6pftU

[You20] *When is Requirements Engineering good enough?* YouTube video: https://youtu.be/VTYVxKomOWA

7 Autor

Christof Ebert ist Geschäftsführer der Vector Consulting Services. Er arbeitet in verschiedenen industriellen Aufsichtsgremien, ist Professor in Stuttgart und Paris, Autor mehrerer Bücher und in den Herausgeberkomitees von Zeitschriften wie "IEEE Software" und "Journal of Systems and Software".
Kontakt: @ChristofEbert

COSMIC Examples – What could they mean as an IT project?

Reiner Dumke, Anja Fiegler, Cornelius Wille

University of Magdeburg, Microsoft Germany,
and Technical University of Applied Science Bingen, Germany

Introduction

The following paper uses the software sizing of the COSMIC examples (see https://cosmic-sizing.org/publication-cat/case-studies/) in order to show some "fictive" estimation of typical project control data and metrics. We use the size measurement of COSMIC Function Points as ***CFP*** described in the different examples.

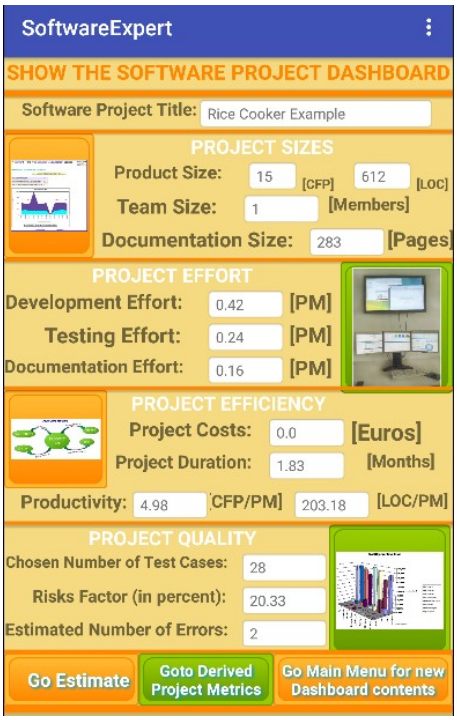
Furthermore, the estimations of the essential project data are based of our ***SoftwareExpert App*** (available in the Google Play Store). The estimation could be select from a list of authors for a special kind of project metrics or you can use all shown metrics building the average values. In our paper we have chosen the average metrics values (the project costs excluded).

COSMIC Examples – Their Sizes and their Dashboard Metrics

In following we describe the COSMIC examples from the COSMIC community websites and our handbook of software estimation. The first examples can be summarized as real-time application software systems (see also [1], [2], [9] and [11]).

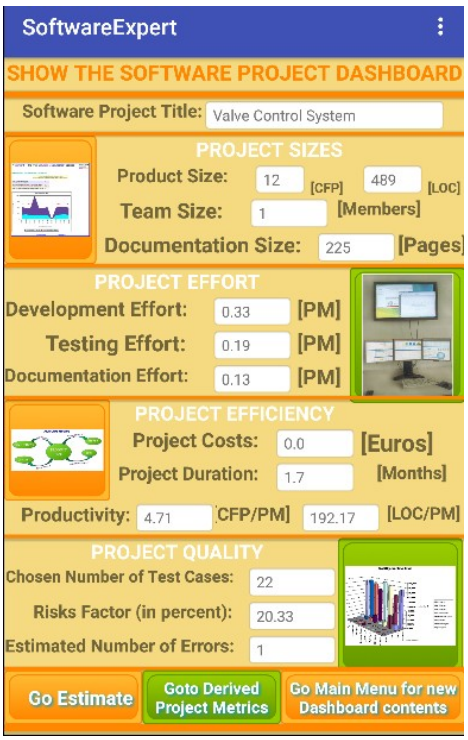
The Rice Cooker Example:

In the *Rice Cooker* software system example are identified four main functional processes (see [1] and [4]): the start cooking with ***3 CFP***, the Update target temperature with ***5 CFP***, the check cooker temperature with ***4 CFP*** and the stop cooking with ***3 CFP***. This small example leads to a total software size of ***15 CFP***.



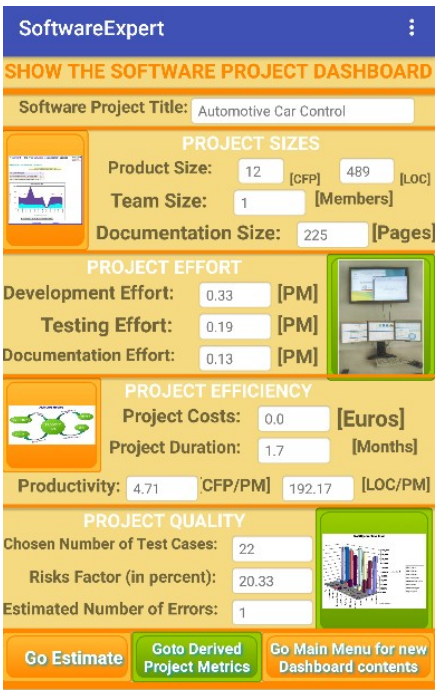
The Valve Control System Example:

The functional process involve the control of time during the operating cycle of the control valve and leads to a total size of ***12 CFP*** [5]***.***



The Automotive Car Control Example:

This example contents the automotive control system with all the embedded software components. Especially, the Servo functional unit has a software size of ***12 CFP*** [10].



The Board Electronic Example:

The whole system of board electronics has more than hundred thousand line of code. Using the COSMIC based conversion rules, we obtain a software size of ***14732 CFP.***

The next examples can be characterized a business applications (see also [9], [10] and [14]).

The Car Hire Example:

This example consists of eight functional processes [6]: the customer list with ***4 CFP,*** the view customer summary with ***5 CFP,*** the view customer details with ***3 CFP, t***he preview customer details with 3 CFP, the update customer details with ***3 CFP,*** the display invoice print preview with ***6 CFP,*** the add of new customer with ***3 CFP*** and the view customer booking details with ***6 CFP.*** This lead to a total size of ***33 CFP.***

The Course Registration Example:

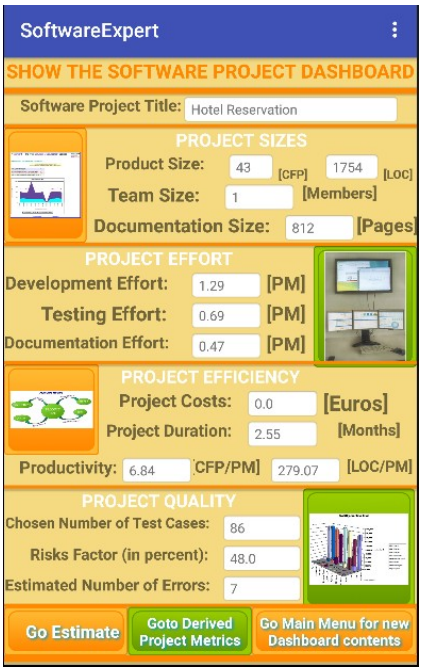
This system example requirements below describe the functionality of the software to be developed by a project that will replace the existing Course Registration System (CRS) with an on-line system (“C-Reg”) that allows students and professors access through PC clients [15]. This system includes 19 functional processes and leads to a total number of ***102 CFP.***

The Machine Learning Example:

In Machine Learning, a neural network is a software application that can 'learn' to classify input data with the help of 'training examples' of that input data [3]. The considered example has seven functional processes and a total number ***39 CFP.***

The Hotel Reservation Example:

The example of a hotel reservation software system includes six functional processes and has a total size number of ***43 CFP*** (an example of Abran described in [10].

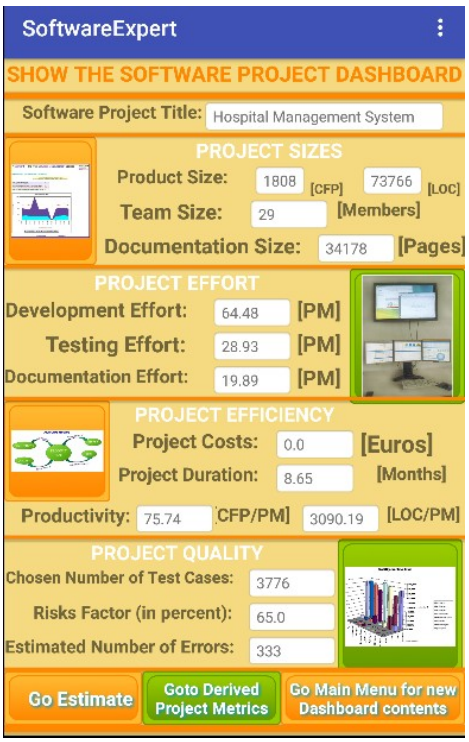


The Web Advice Module Example:

The Web Advice Module is a special module on the website of a commercial bank to assist (young) customers with the choice whether they are going to rent a house or buy one with a mortgage (see [16]). The system consists of nine functional processes and summarizes ***40 CFP.***

The Hospital Management System Example:

The hospital management software system considers 12 functional processes (see [10] and [13]) and leads to ***1808 CFP*** as total number of COSMIC function points.



The Cloud Computing Example:

This example is based on a real implementation of a cloud system at German Telekom and has four large components [described in [10]): the interaction layer with ***9657 CFP,*** the process layer with ***5843 CFP,*** the functional layer with ***13662 CFP*** and the operational layer with ***15500 CFP.*** Hence, ***t***he whole system has ***44662 CFP.***

Summary of the Measurement and Estimation Results

Of course, it is not a surprise that small examples leads to small project data and large ones to big data. Furthermore, the examples have different motivations and intentions. But, some measurements could be interesting and meaningful. Note, the size based estimation don't differ between the kind of system (real-time or business etc.).

In following we show some tables and leave the interpretation to the reader for his own needs (PM means personal month).

COSMIC Example	Team size	Documentation size (in pages)	Lines of Code
Rice Cooker	1	283	612
Valve Control System	1	225	489
Automotive Car Control	1	225	489
Board Electronic	233	278506	60106
Car Hire	1	622	1346
Course Registration	2	1927	4161
Machine Learning	1	736	1591
Hotel Reservation	1	812	1754
Web Advice Module	1	755	1632
Hospital Management System	29	34178	73765
Cloud Computing	707	844330	18222

Tab. 1: Team size and software estimation

Note, our estimations are “classical”. We don’t consider “documentation less” software implementation. Furthermore, most of the examples are very small, which better helps to explain the COSMIC method

COSMIC Example	Dev. Effort in PM	Test Effort in PM	Documentation Effort (in PM)
Rice Cooker	0.42	0.24	0.16
Valve Control System	0.33	0.19	0.13
Automotive Car Control	0.33	0.19	0.13
Board Electronic	525.44	235.71	162.05
Car Hire	0.97	0.53	0.36
Course Registration	3.54	1.63	1.12
Machine Learning	1.16	0.62	0.43
Hotel Reservation	1.29	0.69	0.47
Web Advice Module	1.2	0.64	0.44
Hospital Management System	64.48	28.93	19.89
Cloud Computing	1592.95	714.59	491.28

Tab. 2: Effort estimations

It is interesting to see whether the actual effort required for testing and documentation is actually taken into account.

COSMIC Example	Project Duration in month	Number of Test Cases	Maintenance Effort (in PM)
Rice Cooker	1.83	28	0.84
Valve Control System	1.7	22	0.66
Automotive Car Control	1.7	22	0.66
Board Electronic	17.62	30786	1050
Car Hire	2.34	65	1.94
Course Registration	3.36	212	7.08
Machine Learning	2.47	75	2.32
Hotel Reservation	2.55	86	2.58
Web Advice Module	2.49	82	2.4
Hospital Management System	8.65	3776	128.9
Cloud Computing	25.84	93340	3185

Tab. 3: Project duration and further characteristics

In addition to component, integration and acceptance testing, the test cases also take into account quality aspects such as security, performance and reliability.

COSMIC Example	Estimated Number of Errors	Errors per Team Member	Verification Effort (in PM)
Rice Cooker	2	2	1.7
Valve Control System	1	1	1.36
Automotive Car Control	1	1	1.36
Board Electronic	2717	11	1670
Car Hire	5	5	3.74
Course Registration	18	9	11.56
Machine Learning	6	6	4.42
Hotel Reservation	7	7	4.88
Web Advice Module	7	7	4.54
Hospital Management System	333	11	204.9
Cloud Computing	8238	11	5063

Tab. 4: Quality assurance estimation

In particular, the use of formal test methods helps in quality assurance. However, the effort involved should not be underestimated.

References

1. Abran, A.; Desharnais, J.-M.; Lesterhuis, A.; Symons, C. R.: *Guideline for Sizing Real-time Software v2.0*

2. Abran, A.; Khelifi, A.; Symons, C. R.: *Automatic Line Switching v1.1*

3. Abran, A.; Lesterhuis, A.: *Sizing software in a Machine Learning context: A COSMIC Case study v1.0.* October 2019

4. Abran, A.; Lesterhuis, A.; Symons, C.R.; Trudel, S.: *Rice Cooker Case Study v2.0.1.* August 2018

5. Abran, A.; O'Neill, M.; Khelifi, A.; Roy, C.; Symons, C. R.: *Valve Control Case Study v1.0.1.* August 2018

6. Downing, M.; Eagles, M.; Hope, P.; James, Ph.: *ACME Car Hire Case Study v1.0.1,* August 2018

7. Dumke, R.: *The SoftwareExpert App.* Software Measurement News, 25(2020)1, pp. 5 – 6 and 24(2019)2, pp. 35 – 41 (see in the Google App Store)

8. Dumke, R.: *The SoftwareLite App.* Software Measurement News, 24(2019)2, pp. 30 – 34 (see in the Google App Store)

9. Dumke, R.; Abran, A.: *COSMIC Function Points – Theory and Advanced Practices.* CRC Press, Boca Raton, 2011

10. Dumke, R.; Schmietendorf, A.; Seufert, M.; Wille, C.: *Handbuch der Softwareumfangsmessung und Aufwandschätzung.* Logos-Verlag, Berlin, 2014

11. Lesterhuis, A.: *Industrial Automation (Robot) Case Study.* 2011

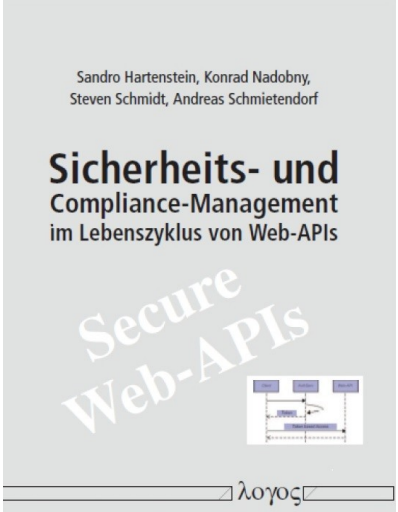
12. Lind, K.; Heldal, R.; Harutyunyan, T.; Heimdahl, T.: *CompSize: Automated Size Estimation of Embedded Software Components.*

13. Schoedon, M.: Hospital Management *System Design and COSMIC FP based Effort Estimation.* Diploma Thesis, University of Magdeburg, 2005

14. Sellami, A.; Haoues, M.; Ben-Abdallah, H.: *Sizing Natural Languages/ User Stories/ UML Use Cases for Web and Mobile Applications using COSMIC FSM.* May 2019

15. Symons, C. R. et al: *Course Registration ('C-REG') System Case Study.* August 2018

16. Vogelezang, F.; Symons, C. R.; Lesterhuis, A.: *Web Advice Module version v1.1.1.* Nesma 2018

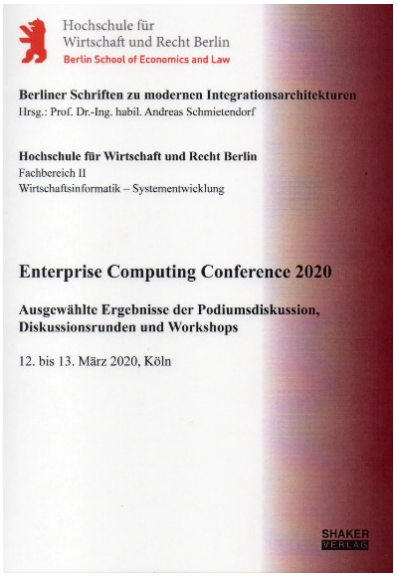


Hartenstein/Nadobny/Schmidt/ Schmietendorf:

Sicherheits- und Compliance Management

Logos-Verlag, Berlin, 2020
ISBN 978-3-8525-5086-8

This book describes approaches and techniques for implementing Web APIs keeping security-related requirements. The API management involves analytical and constructive approaches for quality assurance during the development. The DevOps approach was considered in the context of business processes.



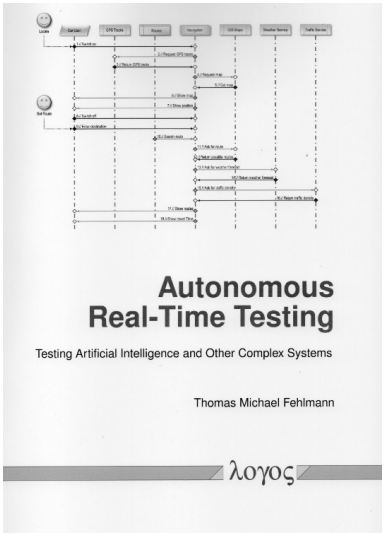
Schmietendorf, A.:

Enterprise Computing Conference 2020

Köln, März 2020

Shaker Verlag, Aachen, 2020,
ISBN 978-3-8440-7320-1

Dieses Buch beinhaltet die Beiträge zur ECC-Konferenz 2020 zur Thematik „Enterprise Transformation“ vor allem in relevanten Anwendungsbereichen.



Thomas M. Fehlmann:

Autonomous Real-Time Testing

Testing Artificial Intelligence and Other Complex Systems

Logos-Verlag, Berlin, 2020
ISBN 978-3-8525-5086-8

The book explains the theory and the implementation approach for a framework for Autonomous Real-Time Testing (ART) of a software-intense system while in operation. Principles and approaches like Combinatory logic,Analytic Hierarchy Process (AHP) and Quality Function Deployment (QFD) are used for a complex testing approach of real-time systems like automotive solutions, IoT control software and embedded system

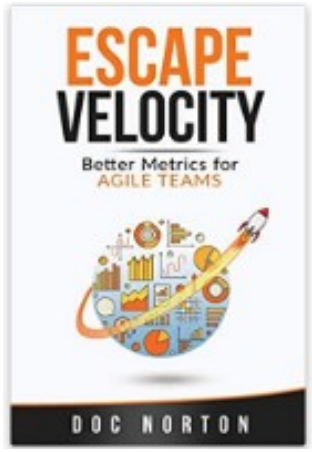
releases.

Andreas Schmietendorf

Empirische Untersuchungen zum Cloud-Einsatz im KMU-Bereich - eine zusammenfassende Betrachtung

Shaker-Verlag, Aachen, April 2020, ISBN 978-3-8440-7356-0

Das vorliegende Buch reflektiert die Ergebnisse von forschungs- aber auch industrieorientierten Projekten rund um die Themenstellung des Cloud Computings, die durch den Autor initiiert und in den vergangenen 10 Jahren verantwortet bzw. im Rahmen seiner Arbeitsgruppe bearbeitet wurden. n, testen und bereitstellen.



Doc Norton

Escape Velocity: Better Mertrics for Agile Teams

Februar 2020

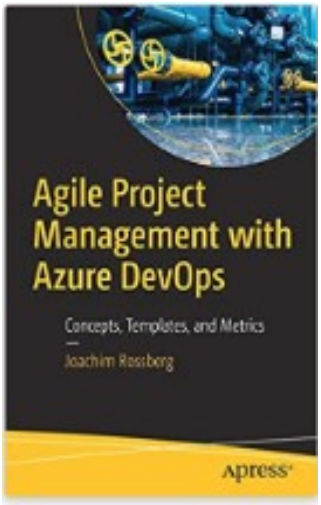
This book identifies the velocity as the most commonly used metric in agile software delivery. The efficiency of Scrum teams is the main focus. Metrics are considered in general and further measure like lead time, team joy, team performance etc. are proposed especially. The book includes many interesting stories of agile team management.

Elyjoy Muthoni Micheni

Metrtics and Models for Evaluating the Quality and Effectiveness of ERP Software

Juli 2019

This book presents a set of theoretical measurement models and metrics for measuring software size and complexity of large scale enterprise resource planning software based on practical experiences. It focuses on the measurement of usability, service quality, security, interoperability, maintenance and enterprise resource planning.



Joachim Rosberg

Agile Project Management with Azure DevOps: Concepts, Templates, and Metrics

April 2019

This book considers Agile project management to use and customize Microsoft Azure DevOps. The basic process involves the Application Life Cycle Management approach and achieve an overall higher quality output.“

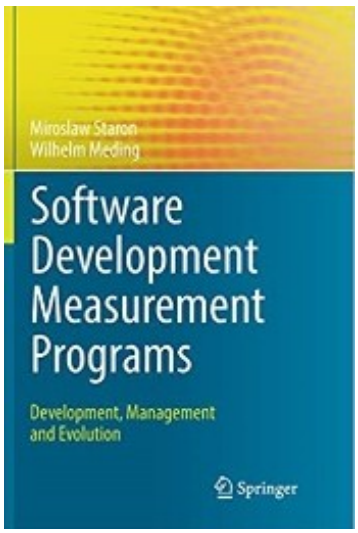
Schmietendorf, A.:

Workshop ESAPI 2019

Dresden, November 2019

Shaker Verlag, Aachen, 2019,
ISBN 978-3-8440-6837-5

Dieses Buch beinhaltet die Beiträge zur ESAPI-Konferenz 2019 zu Sicherheits- und Complianceaspekten von Web-APIs vor allem in relevanten Anwendungsbereichen.



Miroslaw Staron:

Software Development Measurement Programs

Springer Publ., 2019
ISBN 978-3030063085

This book describes approaches and techniques for implementing software measurement processes from a practical point of view involving toll support, project integration and measurement programs evolution.

Schmietendorf, A.,:

ESAPI 2018

2. Workshop: Evaluation of Service-APIs
8. November 2018, München

Shaker Verlag, Aachen, April 2018, ISBN 978-3-8440-6254-0

The book includes the proceedings of the Evaluation of Service-APIs 2018 Workshop held in Munich in November 2018, which constitute a collection of theoretical studies in the field of measurement and evaluation of service oriented and API technologies.

Gerardus Blokdyk:

Software Measurement the Ultimate Step-By-Step Guide

5starcooks Publ. 2018

This book summarizes some helpful practical experiences about measurement integration in software management processes and their successful implementation.

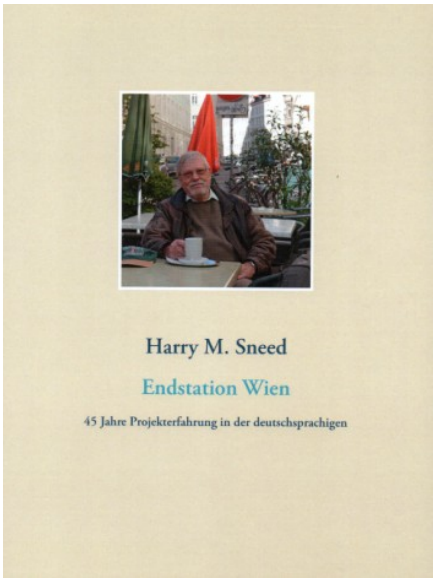
Schmietendorf, A., Nitze, A.:

ESAPI 2018

2. Workshop: API-First/API-Management
19. April, Hamburg

Shaker Verlag, Aachen, April 2018, ISBN 978-3-8440-5927-4

The book includes the proceedings of the API-First/API-Management 2018 Workshop held in Hamburg in April 2018, which constitute a collection of theoretical studies in the field of measurement and evaluation of service oriented and API technologies.



Harry Sneed:

Endstation Wien

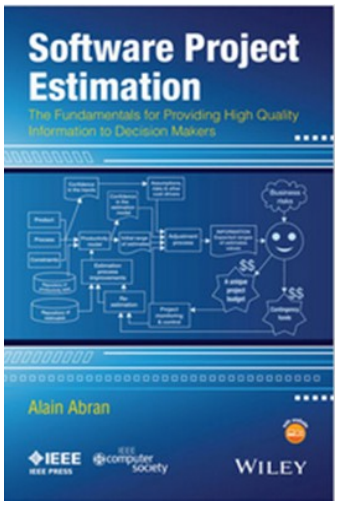
45 Jahre Projekterfahrungen in der deutschsprachigen IT-Welt
BoD Norderstedt, 2017, 328 S.
ISBN 978-3-7448-8364-1

Dieses Buch beschreibt nahezu die gesamte Tätigkeit von Harry Sneed in der IT-Welt, von den Anfängen der Großrechner mit den COBOL und PL/1-Programmen bis hin zu den aktuellen und modernen Ansätzen Service-orientierter Tech-nologien und Systemen. Dieses Buch fasst vor allem die umfangreichen Erfahrungen zu Wartungs-, Migrations- und Testprojekten zusam-men, die auch für die Beherrschung aktueller und moderner Software-Anwendungen, wie beispiels-weise autonomes Fahren oder Smart Cities, von unschätzbarem Wert sind. Es zeigt in unter--haltener Weise die teilweise immensen Anstren-gungen für die kontinuierliche Gewährleistung von IT-Diensten.

Staron, M, Melding, W.:

Proceedings of the IWSM/Mensura 2017

Joined Conference of the 27th International Workshop on Software Measurement (IWSM) and the 12th International Conference on Software Process and Product Measurement (Mensura), ACM 2017, *ISBN 978-1-4503-4853-9*
This proceedings are available at the Computer Science Bibliography of Trier.



Abran, A.:

Software Project Estimation: The Fundamentals for Providing

High Quality Information to Decision Makers

Wiley IEEE Computer Society Press, 2015 (288 pages), ISBN 978-1-118-95408-9

This book introduces theoretical concepts to explain the fundamentals of the design and evaluation of software estimation models. It provides software professionals with vital information on the best software management software out there. End-of-chapter exercises, Over 100 figures illustrating the concepts presented throughout the book, Examples incorporated with industry data.

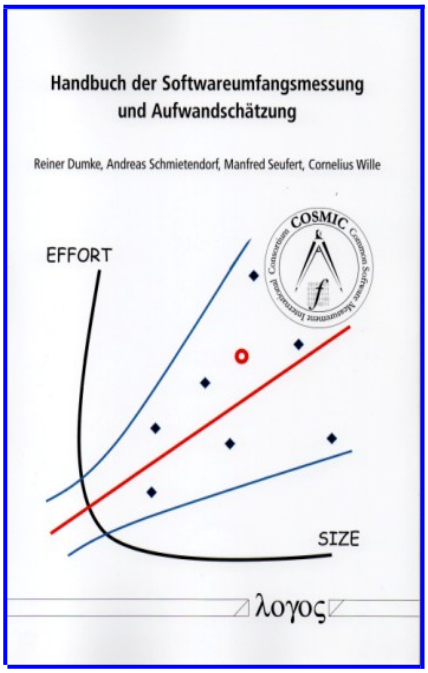
Please remember:

Dumke, R., Schmietendorf, A., Seufert, M., Wille, C.:

Handbuch der Softwareumfangsmessung und Aufwandschätzung

Logos Verlag, Berlin, 2014 (570 Seiten), ISBN 978-3-8325-3784-5

Eine vollständige Beschreibung der COSMIC Function Point Methode mit zahlreichen industriellen Anwendungen und Erfahrungen.



Die Aufwandschätzung von Softwaresystemen ist und bleibt in der heutigen weltweit verzweigten bzw. agilen Softwareerstellung ein aktuelles und vor allem immer bedeutungsvolleres Thema im Bereich der Softwareentwicklung und -bereitstellung. Um den wirklichen Aufwand für eine Softwaresystementwicklung zu bestimmen ist jedoch erst einmal eine möglichst exakte Bestimmung oder gar Messung des zu realisierenden funktionalen Softwareumfangs notwendig. Genau dieser Thematik widmet sich das vorliegende Handbuch in grundlegender Weise. Es behandelt die bekanntesten Umfangsmess- und Aufwandschätzverfahren nach ihrer Evidenz und Effizienz in ihrer praktischen Anwendung.

Den Hauptteil dieses Handbuches bildet eine erstmals umfassende und vollständige Beschreibung der exakten Bestimmung des Softwarefunktionsumfangs nach dem so genannten COSMIC Function Points Ansatz. Diese Methode ist auch unter der Bezeichnung Next Generation of Functional Size Measurement weltweit bekannt und bezieht vor allem moderne Paradigmen und Systemarten, wie die eingebetteten Systeme, das SOA, das Cloud Computing und die Kommunikationssysteme sowie auch moderne Entwicklungsmethoden einer evolutionären oder agilen Softwareerstellung mit ein.

Das vorliegende Handbuch ist sowohl für die Praxis als auch für den auszubildenden Hochschulbereich eine wichtige Grundlage. Es ist für Studenten mit Informatikbezug zu empfehlen, da es sich vor allem dem grundlegenden Problem des Softwareumfangs selbst widmet, welches wiederum für Anwendungen zur Aufwandschätzung aber auch für Komplexitäts- und Risikoanalysen relevant ist. Für den IT-Bereich unterstützt es das Management bei der Auswahl, Bewertung und geeigneteren Anwendung von Methoden der Softwareumfangsmessung und den damit verbundenen Fragen des Aufwandes, der Qualität und der Effizienz. Darüber hinaus werden die involvierten Communities vorgestellt und deren Rolle für die Thematik dieses Handbuch exemplarisch herausgestellt.

Logos Verlag Berlin ISBN 978-3-8325-3784-5

This book shows an overview about the current software size measurement and estimation approaches and methods. The essential part in this book gives a complete description of the **COSMIC measurement method**, their application for different systems like embedded and business software and their use for cost and effort estimation based on this modern ISO size measurement standard.

Software Measurement & Data Analysis Addressed Conferences

August 2020

icABCD'20:	**International Conference on Advances in Big Data, Computing and Data Communication System** August 6 - 7, 2020, Durban, South Africa see: https://10times.com/icabcd
ICDSE 2020:	**International Conference on Data Science and Engineering** August 12 - 14, 2020, Kerala, India see: http://icdse.in/
Euromicro DSD/ SEAA 2020:	**Software Engineering & Advanced Application Conference** August 26 - 28, 2020, Portoroz, Slovenia see: https://dsd-seaa2020.um.si/index.htm/
ISBSG Conference	**ISBSG Conference in a virrtual mode** August 28 – September 4, 2020 see: https://www.isbsg.org/it-conference-2020/

September 2020

QEST 2020:	**International Conference on Quantitative Evaluation of Systems** Aufgust 31 – September 3, 2020, Vienna, Austria see: http://www.qest.org/qest2020/index.html
EuroAsiaSPI² 2020:	**European Systems & Software Process Improvement and Innovation Conference** September 9 - 11, 2020, Düsseldorf, Germany see: http://2020.eurospi.net/index.php/about/eurospi-2016
RE 2020:	**IEEE International Requirement Engineering Conference** August 31 - September 4, 2020, Zurich, Switzerland see: http://re20.org/
SEFM 2020:	**International Conference on Software Engineering and Formal Methods** September 14 - 18, 2020, Amsterdam, Netherlands see: https://event.cwi.nl/sefm2020/
ODSC 2020:	**Open Data science Conference Europe** September 15 - 17, 2020, Dublin, Ireland see: https://www.odsc.com/dublin
ASE 2020:	**Automated Software Engineering** September 21 - 25, 2020, Melbourne, Australia see: https://conf.researchr.org/home/ase-2020
IMMM 2020:	**International Conference on Advances in Information Mining and Management** Sept. 27 – Oct. 01, 2020, Rome, Italy **see: https://www.iaria.org/conferences2020/IMMM20.html**

October 2020

MSR 2020:
Conference on Mining Software Repositories
October 5 - 6, 2020, Seoul, South Korea
see: https://2020.msrconf.org/

ESEIW 2020:
Empirical Software Engineering International Week
October 5 - 9, 2020, Bari, Italy
see: https://eseiw2020.di.uniba.it/

ESEM 2020:
Conference on Empirical Software Engineering and Measurement
October 8 – 9, 2020, Bari, Italy
see: https://eseiw2020.di.uniba.it/esem_conf/

ICSE 2020:
42th International Conference on Software Engineering
October 5 - 11, 2020, Seoul, South Korea
see https://2020.icse-conferences.org/

API 2020:
API Conference 2020
October , 2020, Berlin, Germany
see: https://blog.hwr-berlin.de/schmietendorf/

data2day 2020:
Konferenz für Big Data, Data Science und Machine Learning
October 19 – 21, 2020, Heidelberg, Germany
see: https://www.data2day.de/

CLOUD 2020:
IEEE International Conference on Cloud Computing
October 19 -23, 2020, Beijing, China
see: https://conferences.computer.org/cloud/2020/

ICWS 2020:
International Conference on Web Services
October 19 -23, 2020, Beijing, China
see: https://conferences.computer.org/icws/2020/

SERVICES 2020:
IEEE World Congress on Services
October 19 -23, 2020, Beijing, China
see: https://conferences.computer.org/services/2020/

ICSEA 2020:
International Conference on Software Engineering Advances
October 18 -22, 2020, Porto Portugal
see: https://www.iaria.org/conferences2019/ICSEA19.html

ICST 2020:
IEEE International Conference on Software Testing, Verification & Validation
October 24 - 28, 2020, Porto, Portugal
see: http://icst2020.info/

ASQT 2020:
Arbeitskonferenz Softwarequalität, Test und Innovation
October 29 , 2020, Bozen, Austria
see: http://www.asqt.org/

IWSM/Mensura 2020:
Common International Conference on Software Measurement
October 29 - 30, 2020, Mexico City, Mexico
see: https://www.iwsm-mensura.org/

November 2020

ESEC/FSE 2020:	**European Software Engineering Conference** and **Symposium on the Foundation of Software Engineering** November 6 - 13, 2020, Sacramento, USA see: /https://2020.esec-fse.org/
IEEE ICDM 2020:	**IEEE International Conference on Data Mining** November 17 - 20 , 2020, Sorrento, Italy see: http://icdm2020.bigke.org/
BigDataSE 2020:	**14th IEEE International Conference on Big Data Science and Engineering** November 10-13, 2020, Guangzhou, China see: http://www.ieee-trustcom.org/BigDataSE2020/
PROFES 2020:	**International Conference on Product Focused Software Process Improvement** November 25 -27, 2020, Turin, Italy see https://softeng.polito.it/profes2020/
SERA 2020:	**IEEE/ACIS Conference on Software Engineering Research, Management and Applications** November 30, 2020, Kanazawa, Japan see: https://www.myhuiban.com/conference/943

December 2020

Big Data 2020:	**IEEE International Conference on Big Data** December 10-13, 2020. Altanta, USA see: http://bigdataieee.org/BigData2020/
BCD 2020:	**International Conference on Big Data, Cloud Computing, and Data Science Engineering** December 14 - 16, 2020, Macao see: http://http://acisinternational.org/conferences/bcd-2020/

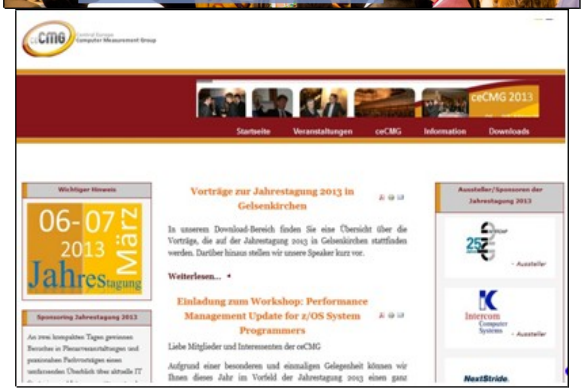
see also:

- http://www.acisinternational.org/newconferences.html
- https://www.acm.org/conferences
- https://www.ieee.org/conferences_events/index.html

COMMUNITIES

Common Software Measurement International Consortium (COSMIC)

http://cosmic-sizing.org



Central Europe Computer Measurement Group (ceCMG)

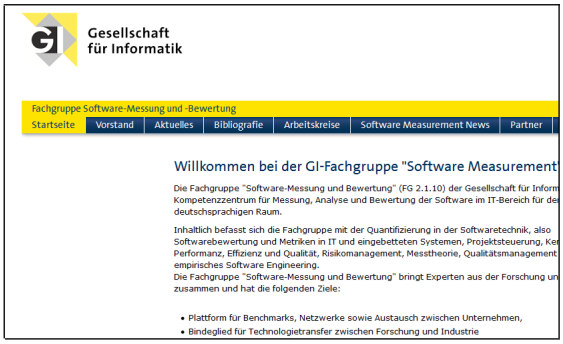
http://www.cecmg.de

Metrics Association's International Network (MAIN)

http://*www.mai-net.org*

Netherlands Software Metrics users Association (NESMA)

http://www.nesma.org/



GI-Fachgruppe Software-Messung und Bewertung

https://fg-metriken.gi.de/

(Measurement News Online)

Deutschsprachige Anwender-gemeinschaft für Software-Metrik und Aufwandschätzung

http://www.dasma.org

International Software Bench-marking Standard Group (ISBSG)

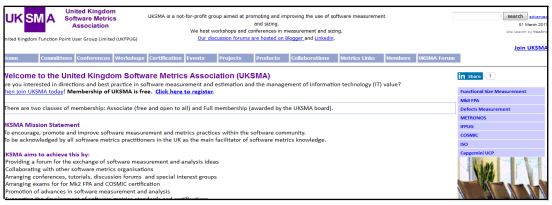
https://www.isbsg.org

Finnish Software Measurement Association (FISMA)

http://www.fisma.fi/in-english/

Asociacion Espanola de Metricas de Software

http://www.aemes.org/



United Kongdom Software Metrics Association (UKSMA)

http://www.uksma.co.uk

Gruppo Utenti Function Point Italia - Italian Software Metrics Association (GUFPI - ISMA)

http://www.gufpi-isma.org

Anwenderkonferenz Software-qualität und Test (ASQT)

http://www.asqt.org

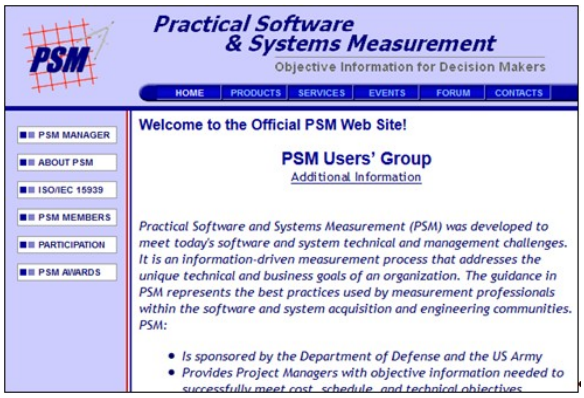
MEASUREMENT SERVICES

Software Measurement Laboratory (SML@b)

http://www.smlab.de

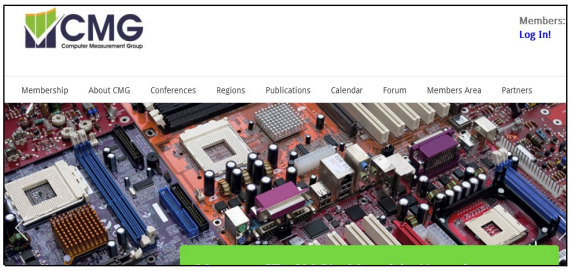
International Function Point Users Group (IFPUG)

http:www.ifpug.org



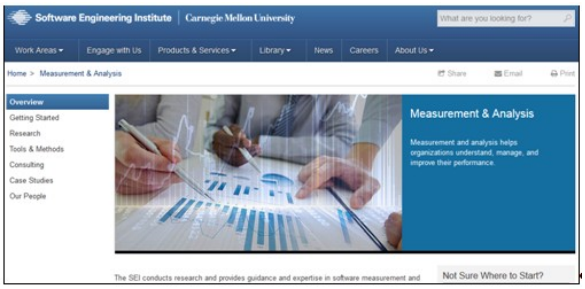
Practical Software & Systems Measurement

www.psmsc.com/:



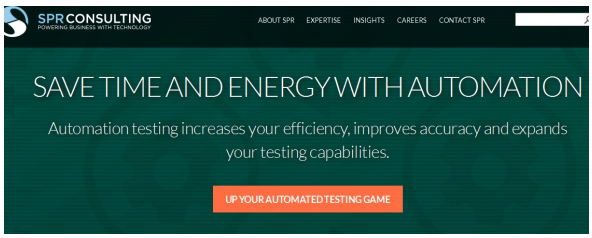
Computer Measurement Group (CMG)

http://www.cmg.org



Software Engineering Institute (SEI)

www.sei.cmu.edu/measurement/



Software Productivity Research (SPR)

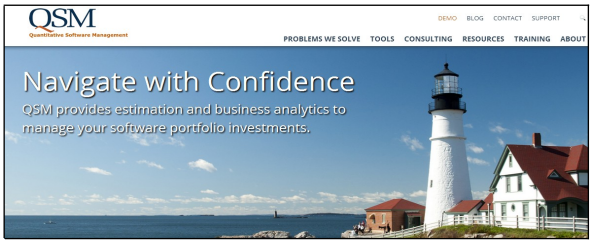
http://www.spr.com/

McCabe & Associates

http://www.mccabe.com

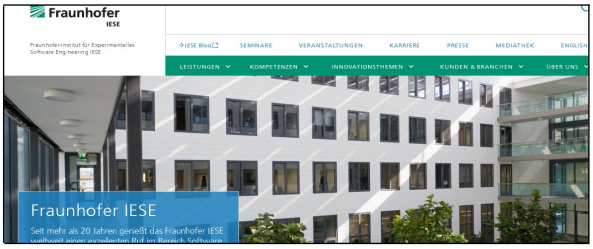
SQS Gesellschaft für Software-Qualitätssicherung

http://www.sqs.de



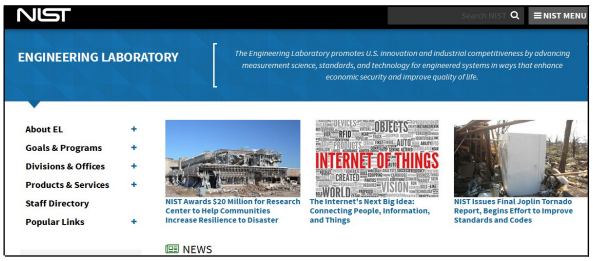
Quantitative Software Management (QSM)

http://www.qsm.com/



Fraunhofer Institute for Experimental Software Engineering (IESE)

https://www.iese.fraunhofer.de/



National Institute of Standards and Technology (NIST)

https://www.nist.gov/el

SOFTWARE MEASUREMENT INFORMATION

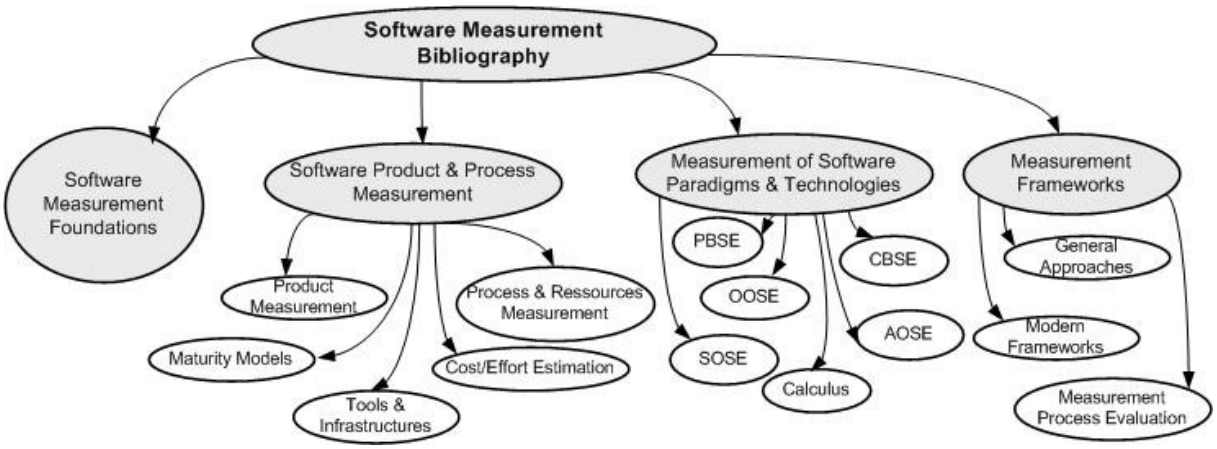
Software Measurement Bibliography

See our overview about software metrics and measurement in the Bibliography at

https://fg-metriken.gi.de/bibliographie/

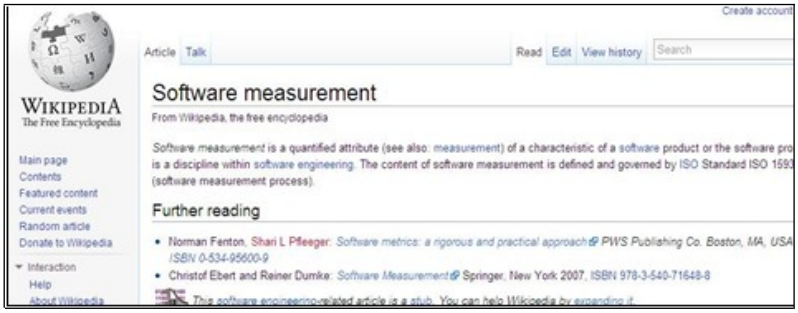
including any hundreds of books and papers

Bibliography Structure:



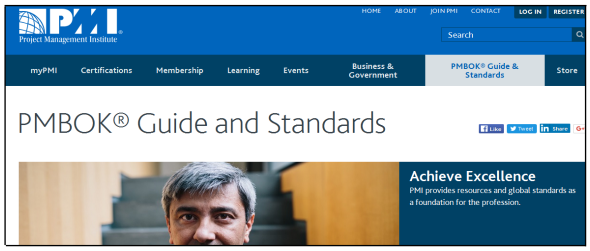
Software Measurement & Wikipedia

Help to qualify the software measurement knowledge and intentions in the world wide web:



Software Engineering Body of Knowledge (SWEBOK)

http://*www.swebok.org*



Project Management Body of Knowledge (PMBOK)

http://www.pmbook.org

SOFTWARE MEASUREMENT NEWS

VOLUME 25 2020 NUMBER 2

CONTENTS

Announcements 2

IWSM Mensura Conference 2020 2

ESAPI Workshop 2020 7

Announcements of the GUFPI-ISMA 11

Announcements of the NESMA 12

Announcements of the GI FG 2.1.10 13

Community Reports 14

News Papers 17

Charles Symons:
Why COSMIC functional Measurement? 17

Reiner Dumke, Anja Fiegler, Cornelius Wille:
COSMIC Examples – What could they mean as an IT project? 20

Caper Jones:
IFPUG Function Point Measurements 31

Christof Ebert:
Test-Orientiertes Requirement Engineering 43

New Books on Software Measurement 53

Conferences Addressing Measurement Issues 59

Metrics in the World-Wide Web 62

ISSN 1867-9196