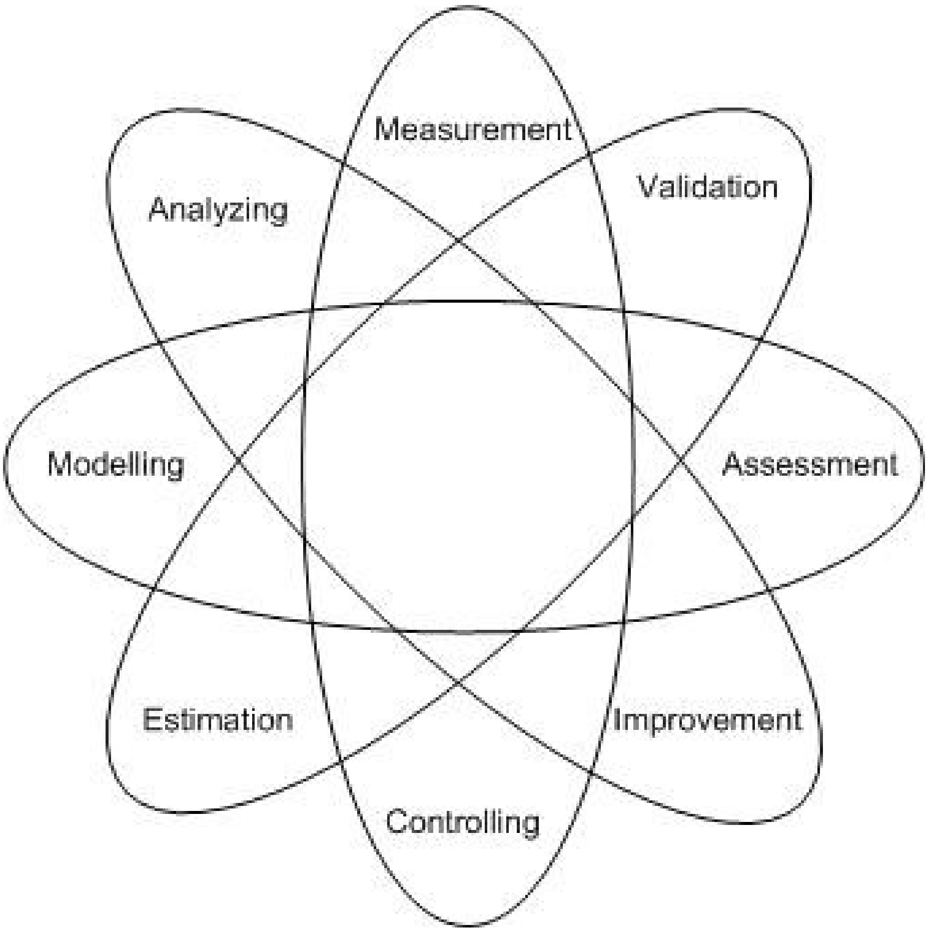
Volume 28, Number 2, October 2023

Software Measurement News

Journal of the Software Measurement Community



Editors:

Alain Abran, Jens Heidrich, Reiner Dumke, Andreas Schmietendorf

Université du Québec
École de technologie supérieure

CONTENTS

Editors:

Alain Abran
Professor and Director of the Research Lab. in Software Engineering Management
École de Technologie Supérieure, 1100 Notre-Dame Quest, Montréal, Quebec, H3C 1K3, Canada, **alain.abran@etsmtl.ca**

Jens Heidrich
Fraunhofer IESE
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
Jens.Heidrich@iese.fraunhofer.de

Reiner Dumke
Professor on Software Engineering,
University of Magdeburg, Faculty of Informatics, Germany,
reiner.dumke@t-online.de, http://www.smlab.de

Andreas Schmietendorf
Hochschule für Wirtschaft und Recht
Alt-Friedrichsfelde 60 10315 Berlin, Germany,
andreas.schmietendorf@hwr-berlin.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.

KI-Szenarien im Zeitalter von ChatGPT & Co

Öffentlicher Expertenworkshop in Anlehnung an die Themen des IFAF-Forschungsprojekts TAHAI (TrustAdHocAI – Details siehe QR-Code)

Veranstalter: GI-FG “Measurement & Data Science” - Arbeitskreis ESAPI

Ort: Fraunhofer IESE Kaiserlautern – **21. November 2023** 09:00 bis 16:00 Uhr

Vorläufige Agenda der Vormittagssitzung (09:30 bis 12:30 Uhr):

Dr. Andreas Jedlitzschka (Fraunhofer IESE)
Prof. Dr. Andreas Schmietendorf (HWR Berlin/Uni Magdeburg)
Eröffnung und Ziele des Workshops

Prof. Dr. Claudia Nass (HS Mainz/Fraunhofer IESE) – Möglichkeiten des Design Thinking im Diskurs von KI-Szenarien – angefragt

Dr. Peter Münte (Universität Innsbruck) – Fachliche Daten transkribierter Geprächsverläufe im Mediationskontext – in Abstimmung

Prof. Dr. Erik Rodner (HTW Berlin) – Benchmarking von KI-Modellen (Online-Beitrag)

Walter Letzel (TU Berlin) – Konzeptioneller Ansatz zur Analyse und Bewertung von Mediationssitzungen (Online-Beitrag)

Sandro Hartenstein (HWR Berlin) – Prototypische Analyse von KI-APIs für LLMs (Cloud vs. OnPremise)

Vorläufige Agenda der Nachmittagssitzung (13:30 bis 16:00 Uhr):

Diskussion ggf. World Café:
Moderation Dr. Jens Heidrich *(HS Mainz/Fraunhofer IESE)*

Mögliche Themenschwerpunkte:

- Identifikation fachlicher KI-Szenarien,
- Bewertung der Datenqualität für KI,
- Auswahl und Bewertung von KI-Services (APIs),
- Rahmenbedingungen für KI-Experimente.

Prof. Dr. Andreas Schmietendorf *(HWR Berlin/Uni Magdeburg)*
Zusammenfassung und Ausblick

Bem.: Die Teilnahme am Workshop ist kostenfrei, dennoch wird um eine Anmeldung (via Email: **tahai@hwr-berlin.de**) zum Zweck der besseren Organisation gebeten. Ggf. notwendige Änderungen der Agenda sind vornehalten.

SML@b News

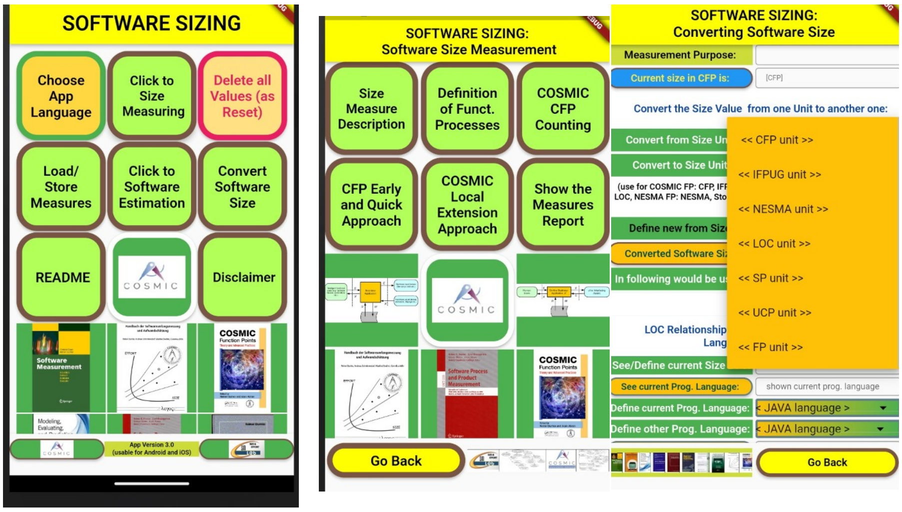
Reiner R. Dumke, University of Magdeburg, Germany

https://softmeasure.de/

Currently, new versions of the SML@b Apps ***SoftwareSizing*** (Software sizing based on the COSMIC FP method) and ***SoftwareCrash*** are available. They are implemented in Flutter/Dart with their typical layouts. These apps are helpful for computer science education and motivation. The apps are free in the Google Play Store.

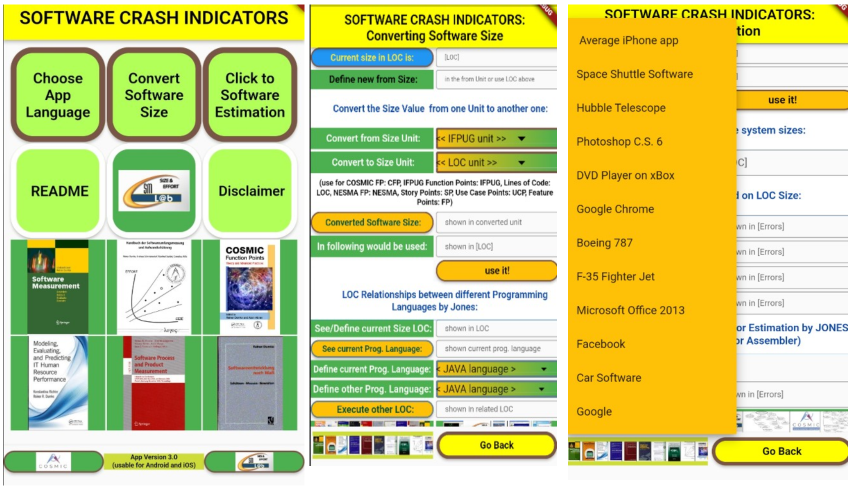
SoftwareSizing

SoftwareSizing helps to accurately measure the software scope. This app serves mainly for the application of the COSMIC Function Point method (as International Standard ISO/IEC 19761). This app also enables the application of a COSMIC method as Early & Quick Method and the local extension approach of the process CFPs as Extend Method. In addition, the app enables conversion between the various software scope measures used in practice, such as the IFPUG Function Points, the Use Case Points, the Story Points and, of course, the Lines of Code, as well as other size metrics.



SoftwareCrash

SoftwareCrash helps to identify software dangers using the software size as lines of code and their relationship to the incorrectness. The main reason for this is the estimation of the number of errors based on experience with professionally developed software in the industrial sector. Furthermore, on the basis of already existing very comprehensive software systems, such an error estimation can be carried out retrospectively. For this purpose, the respective code line scopes known from the literature are shown in a menu button. Hence, this App is useful in the agile development and as educational support for computer science students and professionals.



To use our apps ***SoftwareCount*** and ***SoftwareExpert*** on an iPhone, use the usual emulators *IEMU APK, Cider APK, IOSEmus* etc.

Report on the IWSM/Mensura 2023
September 14 – 15, 2023, Rome, Italy

IWSM MENSURA

ROME, SEPTEMBER 14-15, 2023

The Joint Conference of the 32nd International Workshop on Software Measurement (IWSM) and the 17th International Conference on Software Process and Product Measurement (MENSURA), will be held on September 14-15, 2023 in Rome, Italy.

https://www.iwsm-mensura.org

Conference Abstracts:

An Analysis of Students' Teams Scores at the 2022 Software Estimation Challenge

Donatien Koulla Moulla, Jean-Marc Desharnais and Alain Abran.

This paper presents an analysis of the 2022 edition of the Software Estimation Challenge' organized by the COSMIC Group. The challenge is based on best practices in software effort estimation, including the use of the COSMIC – ISO 19761 standard for sizing software requirements and the early sizing of software functional and non-functional requirements allocated to software functions. The three major components of this challenge consist of sizing the software requirements of a case study, developing an estimation model, and using it to estimate the development ef-

fort for the case study provided. While a previous study was based on a survey of a sub-group of 22 teams who participated in the 2022 edition of this challenge, the study reported here is based on the analysis of students' team scores across teams and contexts of participation. To help the teams' tutors and students plan and prepare for future challenges, this study presents an analysis of how teams performed across each of the challenge tasks. In summary, the teams performed best in the tasks limited to the application of preprogrammed statistical formula, and much worse in tasks requiring analytic skills for the sizing of the requirements or in the application of their reasonably well-built estimation model to the practical case study they had sized (that is, moving from theory to practice).

Functional Size Measurement for X86 Assembly Programs

Donatien Koulla Moulla, Abdel Aziz Kitikil, Ernest Mnkandla, Hassan Soubra and Alain Abran.

Functional size measurement (FSM) provides a reliable and objective way to measure productivity and estimate the effort required for software activities. FSM automation, compared to manual measurement, enables to reduce errors, increase consistency, and improve the efficiency of the measurement process. This paper presents a COSMIC-based automated FSM for X86 assembly programs widely used in personal computers and servers. The proposed approach applies the COSMIC-ISO 19761 rules for sizing Functional User Requirements (FUR) of an X86 assembly program by identifying functional processes, data groups and data movements. An automated measurement prototype tool is also presented including its architecture, its measurement algorithm, the verification protocol used and its measurement accuracy on the measurement of two assembly programs. The prototype tool can be useful to organizations and practitioners in the embedded systems industry.

AI-based Fault-proneness Metrics for Source Code Changes

Francesco Altiero, Anna Corazza, Sergio Di Martino, Adriano Peron and Luigi Libero Lucio Starace.

In software evolution, some types of changes to the codebase (e.g.: a local variable renaming refactoring) are less likely to introduce faults than others (e.g.: changes involving control flow statements). Effectively estimating the fault-proneness of codebase changes can provide a number of advantages in the software process. For ex-

ample, expensive and time-consuming regression testing, code review, or fault localization activities could be driven by fault-proneness, prioritizing the most critical changes to detect issues more rapidly. A number of works in the literature have focused on predicting the fault-proneness of software systems. Less work, however, has focused on the fault-proneness of evolutionary changes to a codebase, and existing approaches typically require project-specific historical data to be used effectively. This paper presents a set of AI-based metrics designed to estimate the fault-proneness of source code changes. The proposed metrics are based on Tree Kernel functions and Transformer models, that have been largely and effectively used in the Natural Language Processing domain. Moreover, the proposed metrics can be used on any software project, and do not require fine-tuning with project-specific historical data. The effectiveness of the proposed metrics is assessed by applying them to a dataset of real-world source code evolution scenarios, and by comparing them against fault-proneness scores provided by a Software Engineering practitioner. Results are promising and show that the proposed metrics are strongly correlated with human-defined fault-proneness scores, and could thus be used as a good proxy of costly human evaluations. The results also motivate further research on the application of these metrics to concrete scenarios such as regression testing.

Software development effort estimation using Function Points and simpler functional measures: a comparison

Luigi Lavazza, Angela Locoro and Roberto Meli.

Background — Functional Size Measures are widely used for estimating the development effort of software. After the introduction of Function Points, a few "simplified" measureshave been proposed, aiming to make measurement simpler and quicker, but also to make measures applicable when fully detailed software specifications are not yet available. It has been shown that, in general, software size measures expressed in Function Points do not support more accurate effort estimation with respect to simplified measures. Objective — Many practitioners believe that when considering "complex" projects, i.e., project that involve many complex transactions and data, traditional Function Points measures support more accurate estimates than simpler functional size measures that do not account for greater-then-average complexity. In this paper, we aim to produce evidence that confirms or disproves such belief. Method — Based on a dataset that contains both effort and size data, an empirical study is performed, to provide some evidence concerning the relations that link functional size (measured in different ways) and development effort. Results — Our analysis shows that there is no statistically significant evidence that

Function Points are generally better at estimating more complex projects than simpler measures. Function Points appeared better in some specific conditions, but in those conditions they also performed worse than simpler measures when dealing with less complex projects. Conclusions — Traditional Function Points do not seem to effectively account for software complexity. To improve effort estimation, researchers should probably dedicate their effort to devise a way of measuring software complexity that can be used in effort models together with (traditional or simplified) functional size measures.

A Method for Metric Management at a Large-Scale Agile Software Development Organization

Pascal Philipp, Franziska Tobisch, Leon Menzel and Florian Matthes.

The benefits of agile methods for small projects inspired organizations to scale these methods to more extensive settings consisting of multiple agile teams. Such scaling agile settings are more complex, which can make maintaining situational awareness difficult. Metrics can alleviate this problem by increasing insight into the development organization. However, adopting metrics comes with various socio-technical challenges, and current research is missing guidance on metric management in large agile organizations. Therefore, we present a goal-based method designed for a large agile case organization to support stakeholders in selecting, operating, and scaling metrics. Moreover, based on the learnings at the case organization, we present design principles that can potentially guide the development of methods suitable for other contexts. We conducted this research following an action design research (ADR) approach combined with situational method engineering (SME). Our findings indicate that our method proved effective for the case organization. This was accomplished by combining well-established elements (e.g., goal-orientation and tool support) from measurement programs designed for traditional software engineering with unique elements of our method (e.g., metric scaling activities and alignment with agile software development). With this study, we provide deep insights into how metrics are managed at a large agile case organization. Researchers and practitioners can use this work as a foundation for designing measurement programs suitable to other scaling agile organizations.

Functional Size Measurement automation for IoT Edge devices

Salma Salem and Hassan Soubra.

The Internet of Things (IoT) was born when Edge devices were interconnected via the internet allowing them to send and receive data back and forth. Edge devices are actually embedded systems processing data in real-time close to the physical environment in which they are deployed. Functional Size Measurement (FSM) is a tool to measure functionality provided by a software and obtain technical indicators early in the design phase. The COSMIC method is an international standard for functional size measurement of a software. A study [2] discussed FSM for Internet of things (IoT) real-time embedded software and proposed a guideline using Arduino. Another study in [8] proposed the idea of Universal IoT metrics and automated some metrics proposed. This paper presents an approach to automatically measure COSMIC functional size of any IoT Edge Device code. This paper is based on the measurement procedure rules presented in [2], and it attempts to complete the tool presented in [8].

Understanding Developer Practices and Code Smells Diffusion in AI-Enabled Software: A Preliminary Study

Giammaria Giordano, Giusy Annunziata, Andrea De Lucia and Fabio Palomba.

To deal with continuous change requests and the strict time-to-market, practitioners and big companies constantly update their software systems to meet users' requirements. This practice force developers to release immature products, neglecting best practices to reduce delivery times. As a possible result, technical debt can arise, i.e., potential design issues that can negatively impact software maintenance and evolution and, in turn, increase both the time-to-market and costs. Code smells—sub-optimal design decisions identifiable by computing software metrics and providing a general overview of code quality —are common symptoms of technical debt. While previous research focused on code smells primarily considering them in the context of Java, the growing popularity of Python, particularly for developing artificial intelligence (AI)-Enabled systems, calls for additional investigations. This preliminary analysis addresses this gap by exploring the diffusion of Python-specific code smells, and the activities performed by developers that induce the introduction of code smells in their systems. To perform our preliminary investigation, we selected 200 AI-Enabled systems available in the Niche dataset; We extracted 10,611 infor-

mation on the releases using PyDriller, and PySmell to extract information about code smells. The results reveal several insights: 1) Code smells related to object-oriented principles are rarely detected in Python; 2) Complex List Comprehension is the most prevalent and the most long-alive smell; 3) The main activities that can induce code smells are evolutionary. This study fills a critical gap in the literature by providing empirical evidence on the evolution of code smells in Python-based AI-enabled systems.

Starting a new REST API project? A performance benchmark of frameworks and execution environments

Sergio Di Meglio, Luigi Libero Lucio Starace and Sergio Di Martino.

REST APIs have become widely adopted in the software industry, finding extensive usage for businesscritical purposes such as data exchange, mobile app development, and microservice architectures. Such popularity has led to a proliferation of dedicated frameworks, making it challenging for developers and organizations to choose which one to use for developing resource-efficient solutions. Adding to the complexity is the possibility of adopting also different execution environments, such as GraalVM, that offer advantages such as faster startup times, lower memory footprint, and polyglot capabilities. Some prior works have investigated the performance of various frameworks for REST APIs. Still, these studies often consider simplistic scenarios with a single endpoint, which fail to capture the complexity and diversity of real-world REST API applications. Furthermore, the impact of different execution environments on performance was often overlooked. Consequently, there remains a significant knowledge gap in comprehensively assessing the combined influence of frameworks and execution environments on the performance of REST APIs. This study aims to move a first step towards bridging that gap, by conducting a thorough performance benchmark that encompasses real-world REST APIs and considers also the effects of different execution environments. More in detail, the study focuses on two of the most popular programming language and framework combinations for REST APIs, namely JavaScript with the Express framework and Java with the Spring framework. As for the execution environment, we consider both mainstream execution environments for JavaScript and Java (Node and OpenJDK, respectively), and GraalVM, which can execute both Java and JavaScript software. The benchmarking process involves conducting realistic load and stress tests using state-of-the-art tools. Results reveal significant differences in performance across the considered combinations, providing insights that could support developers and system architects in making more informed decisions on the technologies to use for their REST API projects.

An Empirical Study on the Performance of Vulnerability Prediction Models Evaluated Applying Real-world Labelling

Giulia Sellitto, Alexandra Sheykina, Fabio Palomba and Andrea De Lucia.

Software vulnerabilities are infamous threats to the security of computing systems, and it is vital to detect and correct them before releasing any piece of software to the public. Many approaches for the detection of vulnerabilities have been proposed in the literature; in particular, those leveraging machine learning techniques, i.e., vulnerability prediction models, seem quite promising. However, recent work has warned that most models have only been evaluated in in-vitro settings, under certain assumptions that do not resemble the real scenarios in which such approaches are supposed to be employed. This observation ignites the risk that the encouraging results obtained in previous literature may be not as well convenient in practice. Recognizing the dangerousness of biased and unrealistic evaluations, we aim to dive deep into the problem, by investigating whether and to what extent vulnerability prediction models' performance changes when measured in realistic settings. To do this, we perform an empirical study evaluating the performance of a vulnerability prediction model, configured with three data balancing techniques, executed at three different degrees of realism, leveraging two datasets. Our findings highlight that the outcome of any measurement strictly depends on the experiment setting, calling researchers to take into account the actuality and applicability in practice of the approaches they propose and evaluate.

Metrics and Models for Developer Collaboration Analysis in Microservice-Based Systems. A Review

Xiaozhou Li, Amr S. Abdelfattah, Ruoyu Su, Joseph Lee, Ernesto Aponte, Rachel Koerner, Tomas Cerny and Davide Taibi.

Microservices enable different teams to develop and deploy services independently. Practitioners are frequently mentioning the need for independence between teams and developers, and the need for metrics to measure developer collaboration. To shed light on the existing metrics and models, we conducted a Systematic Mapping Study to identify models for measuring the development activities, the metrics adopted by these methods, and the output produced by the methods themselves. We identified 10 different models proposed in 14 research papers. Results show that a large amount of the existing models adopt qualitative metrics, questionnaires, and

surveys to collect the information required while others use information from issue tracking and version systems. The results will enable practitioners and researchers to further validate and extend the research on metrics for evaluating the collaboration and independence among developers

Size Measurement and Effort Estimation in Microservice-based Projects: Results from Pakistan

Görkem Kılınç Soylu, Hüseyin Ünlü, Isra Shafique Ahmad and Onur Demirörs.

During the last decade, microservice-based software architecture has been a common design paradigm in the industry and has been successfully utilized by organizations. Microservice-based software architecture, specifically in the form of reactive systems, has substantial differences from the more conventional design paradigms, such as the object-oriented paradigm. The architecture moved away from being data-driven and evolved into a behavior-oriented structure. The usage of a single database is replaced by the structures in which each microservice is developed independently and has its own database. Therefore, adaptation demands software organizations to transform their culture. In this study, we aimed to get an insight into how Pakistani software organizations perform size measurement and effort estimation in their software projects which embrace the microservice-based software architecture paradigm. For this purpose, we surveyed 49 Pakistani participants from different agile organizations over different roles and domains to collect information on their experience in microservice-based projects. Our results reveal that although Pakistani organizations face challenges, they continue using familiar subjective size measurement and effort estimation approaches that they have used for traditional architectures.

A Maturity Model Guidance Approach for Integration Testing of Avionics Software

Gülsüm Güngör and Ayça Kolukısa Tarhan.

Safety-critical software failures lead to serious results such as loss of live or damage to the environment; therefore, safety-critical software verification requires special attention. Avionics system software is one type of safety-critical software. “DO-178C: Software Considerations in Airborne Systems and Equipment Certification” *was released in 2011 by RTCA, Inc., which defines processes for airborne systems software development and verification. On the other hand, there are well-defined guidelines to improve verification and validation processes of software system development, specifically for software testing. TMMI (Test Maturity Model Integration) model is produced by TMMI Foundation as a guidance for organizations to improve their test processes and product quality. Avionics system software has own safety-related software characteristics, and TMMI does not specifically address software testing practices of these characteristics. To fill this gap, we first identify avionics software characteristics from DO-178C handbook as the base for software testing, and then propose a domain specific guidance document that employs TMMI practices as complementary to DO178C activities. The document is aimed to help test organizations in improving test processes by focusing on airborne software characteristics. The proposed approach is targeted for integration testing of avionics software, since this level of testing is very critical for defect prevention in the safetycritical domain.*

Resolving the Historical Confusions about the Meaning of Software Size and Its Use for Project Effort Estimation

Charles Symons.

The software industry does not have a good track record of delivering systems on time and budget. In part this is due to weaknesses in software sizing and project effort estimating methods and practices. This paper gives the author's perspective on how some of these weaknesses have arisen historically, resulting in differing views on some basic underlying concepts that are still causing confusion today. Some remedies are proposed to help eliminate the confusions. It is not easy to define what we mean by the 'size' of an item of software, and this size is not always clearly distinguished from the size of the project or the activity to develop and implement the software item. Even in recent years the Agile community has defined a measure (Story Points) of the size of a User Story (a simple statement of software requirements) but has used the same units of measurement to estimate or measure the effort to develop and implement the Story. A software item manifests itself in different forms as it progresses through the states of its lifecycle from requirements, design, and source code to executing code (using a 'waterfall' development process for ease of illustration). In each state, the software item's artefacts differ thus necessitating different size measures, for example counts of SLOC (Source Lines of Code) for the source code, and bytes for the executing code. Software size is usually the biggest driver of the effort to develop the size. Consequently, designers of *software size measurement methods all aim that their sizes should correlate with effort. Allan Albrecht and other designers of early methods of sizing software requirements proposed to measure size as the product of two factors which became known as: i) the 'Unadjusted Function Point'(UFP) size, which is a measure of the required software functionality, and ii) a 'Value Adjustment Factor' (VAF) that aims to account for the relative difficulty of developing the software, e.g. by considering its 'complexity' and other types of requirements. These various VAF constituents are called 'size-drivers' in this paper. The product of a UFP and a VAF gives a software size in units of 'Function Points' (FP). The paper argues why those software size measures that are one-dimensional (e.g. UFP, SLOC, bytes) may all be thought of as different types of software 'length'. It further concludes that there is no reason to expect that different methods for measuring the 'length' of software items at different states in their life-cycle will produce results that correlate well with each other, or with measurements of the same items that attempt to also account for the difficulty in developing the sizes, or that account for this difficulty in different ways. Effort estimation methods must take account of other factors besides size, referred to as 'effort-drivers' in this paper. These can be either 'pure' effort drivers, i.e. attributes only of the project, such as the numbers and capability of the staff assigned to the project, or other types of requirements for the software system that are not taken into account by the software size measures such as for its quality, e.g. portability requirements, or technical requirements such as the required response time. The paper notes that the designers of different size measurement methods and different effort estimation methods have, at different times, made different decisions on whether to classify some of these factors as size-drivers or as effort-drivers. A further conclusion, therefore, is that accurate effort estimation requires a coherent allocation of these factors as either size-drivers or as effortdrivers. The result will be a 'Coherent Size/Effort Ecosystem'. Not recognising this need for coherence is another cause of confusion and a source of error in effort estimation practices that rely on converting an estimated size from one state, e.g. FPs, to another size, e.g. SLOC, for a different state before converting size to effort. The paper next discusses the weights applied to the various components of the UFP and VAF sizes. These were all derived by expert judgements of the relative effort to develop the components. It is concluded that these early FP sizes are all, strictly-speaking, standard measures (or indices) of Relative Effort for a software project. In spite of this conclusion, the paper also shows that UFP-like sizes conform to the principles of Functional Size Measurement (FSM) for measuring Functional User Requirements (FUR) as defined by ISO/IEC. A UFP size may therefore also be legitimately regarded as a measure of software Functional Size. But VAF-like factors do not conform to these same principles, and this has led to their demise. As a matter arising from this discussion, the paper proposes improvements to the ISO/IEC definition of the term 'FUR' to clarify its real meaning. The* *final major topic discussed is what we mean by the 'Non-Functional Requirements' (NFR) of a software-intensive system or software product. There are various definitions of NFR and varying ways of accounting for them in effort estimation. The different definitions list different examples as types of non-functional requirements; these lists overlap with the list of requirement types that were accounted for by VAF-like measures, and with the requirement types that are given as examples of what are not FUR in the ISO/IEC definition. In response to this variety of definitions, the COSMIC and IFPUG organizations collaborated to agree a definition of NFR which was compatible with the ISO/IEC definition of FUR. A consequence of these two definitions was that any software requirement could be clearly classified as either functional or non-functional, With hindsight, this definition is flawed, as is the ISO/IEC definition of FUR because both define quality requirements as non-functional. This has caused confusion in practice, because many quality requirements appear, when first expressed, to be NFR but on closer examination evolve into FUR for software, or into a mixture of both FUR for software and requirements for 'nonsoftware' e.g. hardware. As an example, a requirement for the security of a software system (typically regarded as a quality requirement) may be met by software or hardware, or a mixture. The paper therefore proposes further revisions to the ISO/IEC definition of FUR and offers two options to revise the COSMIC/IFPUG definition of NFR to help resolve this confusion. These revisions recognise that all requirements for a software system must ultimately be allocated to software or to 'non-software'. The revised definitions leave customers and project teams to decide how quality requirements will be allocated in their own individual developments, regardless of whether they think of the requirements as a FUR or as NFR. The paper then examines a method ('SNAP') which claims to measure a collective size of the NFR for a software item. The author gives several reasons why he considers this approach as an unwise way of dealing with NFR in a size/effort ecosystem. In overall conclusion, the paper identifies two priorities for action by the software metrics community: first, to accept the proposed revisions to the ISO/IEC definition of FUR and to the COSMIC/IFPUG definition of NFR, and second to convince the Agile community to incorporate a modern functional size measurement method into their sizing/estimating eco-systems.*

The ABC Model: coming “back to the future” in (ICT) contracts

Luigi Buglione.

One of the main problems in current ICT projects is to determine the economic value of project activities using mostly (or solely) the deliverables produced, considering such projects as something repeatable and applying the ‘economies of scale’ principles. But looking to the inside of those projects, most of them are “artisanal” projects, unique to a specific customer for a specific need and many variables should be taken into account in order to provide in an estimation the right ‘quantity’ to be produced (also in terms of outcomes), effort and costs (to be translated into a final price) considering all the activities in scope to such project, not only those strictly devoted to produce the project deliverables. This paper will discuss the current situation and a simple but effective solution to such issues using benchmarking and data management best practices, overcoming also bad outsourcing practices, called the ‘ABC Model’, providing an example with objective evidence.

Proceedings Publication:

The IWSM-Mensura-2023 proceedings will be published as CEUR proceedings during the next time.

Summary of the 17th International Conference on Software and System Processes (ICSSP 2023)

Liming Zhu
CSIRO's Data61
Eveleigh, Australia
Liming.Zhu@data61.csiro.au

Sherry Xu
CSIRO's Data61
Eveleigh, Australia
Xiwei.Xu@data61.csiro.au

Jens Heidrich
Fraunhofer IESE
Kaiserslautern, Germany
Jens.Heidrich@iese.fraunhofer.de

Gayan Benedict
University of Technology Sydney
Sydney, Australia
Gayan.Benedict@uts.edu.au

ABSTRACT

The 17th International Conference on Software and System Processes (ICSSP 2023) was held in conjunction with the 45th International Conference on Software Engineering (ICSE 2023) in Melbourne, Australia, May 14-15, 2023. ICSSP is a leading international forum for research on software and systems processes. ICSSP conferences have been organized by the International Software and Systems Process Association (ISSPA) or its predecessor, the International Software Process Association (ISPA). The 2023 conference extended a sequence of predecessor conferences and workshops stretching back more than a decade. For the 2023 ICSSP, the ISSPA was joining forces with representatives of CSRIO's Data61 unit, the data and digital specialist arm of Australia's national science agency.

ICSSP 2023 was focusing on software and systems processes for and with these new technologies, as well as on incorporating these new technologies into new modes of working in software and systems development for business, government, society, and the environment. The conference received 23 submissions of research, experience, and short papers from which 10 high-quality papers (43%) were accepted after the reviewing process. Complementing the technical program, we had five industry talks sharing experiences and a panel discussion. We also have been able to allure three inspiring and very interesting keynote talks.

KEYWORDS

Software development, systems development, development process, emerging technologies, Blockchain, Artificial Intelligence, low-code / no-code, Augmented, virtual, and extended reality.

1 Introduction

ICSSP conferences have been organized by the International Software and

Systems Process Association (ISSPA) or its predecessor, the International Software Process Association (ISPA). The ICSSP series grew out of two earlier and partly overlapping series, the International Conferences on Software Process (ICSP), dating back to 1991, and the International Software Process Workshops (ISPW), dating back to 1984.

Recent ICSSP conferences have mostly been organized with or around ICSE, with proceedings published by ACM or IEEE. The 2021 and 2022 conferences were co-organized with the International Conference on Global Software Engineering (ICGSE), and ICSSP 2022 was co-sponsored and co-organized with the Carnegie Mellon University Software Engineering Institute. For the 2023 ICSSP, the ISSPA was joining forces with representatives of CSRIO's Data61 unit, the data and digital specialist arm of Australia's national science agency.

In recent years, innovative software and systems technologies have been rapidly adopted with increasingly widespread but uncertain impacts. These technologies include various forms of AI; big data and data science; blockchain and distributed ledgers; augmented, virtual, and extended reality; low-code/no-code platforms; and even quantum computing (among others). By their novel nature, these new technologies and applications demand new approaches to software and systems development. Beyond that, the novelty and uncertainty surrounding these technologies and applications requires new and more effective approaches and processes to assure correctness, understandability, predictability, responsibility, security, ethics and other vital qualities of our software and systems.

At the same time, many of these new technologies may have applications within the software life cycle. So, the question naturally arises, can we put technologies such as AI, blockchain, low code, and big data, to work in improving our ability to develop, manage, and apply new software and systems applications. We can even ask whether it will become essential to do so.

ICSSP 2023 was focusing on software and systems processes for and with these new technologies, as well as on incorporating these new technologies into new modes of working in software and systems development for business, government, society, and the environment. ICSSP 2023 aimed to bring together researchers and practitioners to share their research findings, experiences, and new ideas on diverse topics related to software and system processes. The goal was to advance both the state of the research and the state of the practice by applying innovative ideas from different fields of research to the future of software and system engineering processes.

This year's conference received 23 submissions of research, experience, and short papers from which 10 high-quality papers (43%) were accepted after the reviewing process. Complementing the technical program, we had five industry talks sharing experiences and a panel discussion. We also had been able to allure three inspiring and very interesting keynote talks.

Altogether, this international event was an excellent forum for practitioners, academics, and students from around the world to virtually meet and discuss issues related to most critical topics in Software and Systems Processes.

Table 1: **PC Members for ICSSP 2023**

Ove Armbrust	Dan Houston	Tijs Slaats
Alberto Avritzer	Liguo Huang	Viktoria Stray
Dilum Bandara	Hajimu Iida	Stanley Sutton
Reda Bendraou	Jil Klünder	Binish Tanveer
Gayan Benedict	Martin Kowalczyk	Paolo Tell
Ivana Bosnić	Marco Kuhrmann	Eray Tüzün
Ricardo Britto	Xiao Liu	Michael Unterkalmsteiner
Jieshan Chen	Fabrizio Maria Maggi	Muhammad Usman
Paul Clarke	Antonia Mas Pichaco	Aurora Vizcaíno
Chiara Di Francescomarino	Antoni Lluís Mesquida	Hironori Washizaki
Christof Ebert	Juergen Münch	Mairieli Wessel
Michael Felderer	Joyce Nakatumba	Dietmar Winkler
Eduardo Figueiredo	So Norimatsu	Krzysztof Wnuk
Awdren Fontão	Leon Osterweil	Sherry Xu
Steven Fraser	Ali Ouni	Hasan Yasar
Liliana Guzman	Özden Özcan Top	Murat Yilmaz
Jens Heidrich	Dietmar Pfahl	He Zhang
Graham Hellestrand	Stefan Sauer	
James Hoang	Lin Shi	

2 Organization

Organizing committee of ICSSP 2023 consisted of the following people:

- Liming Zhu, General Chair (CSIRO’s Data61, Australia)
- Xiwei (Sherry) Xu, Program Co-Chair (CSIRO’s Data61, Australia)
- Jens Heidrich, Program Co-Chair (Fraunhofer IESE, Germany)
- Gayan Benedict, Industry Chair (University of Technology Sydney, Australia)
- Junjie Wang, Diversity Chair (Institute of Software at Chinese Academy of Sciences and University of Chinese Academy of Sciences, China)
- Sung Une (Sunny) Lee, Local Chair (CSIRO's Data61, Australia)
- Yin Kia Chiam, Publicity Co-Chair (University of Malaya, Malaysia)
- Rifat Ara Shams, Publicity Co-Chair (CSIRO's Data61, Australia)
- Dawen (David) Zhang, Website and Proceedings Chair (CSIRO's Data61 & Australian National University, Australia)

The program committee had 55 members from 19 countries around the world. They are listed in Table 1.

3 Submissions, Acceptance, and Attendance

ICSSP 2023 received 23 paper submissions including 15 full papers (up to 11 pages), 4 short papers (up to 6 pages), 4 experience reports (up to 6 pages). In addition, we received two industry talk submissions. All submissions were reviewed by our 55 PC members, who overall conducted 77 reviews. At the end of the review period, we had an intense discussion about the results.

Finally, we decided to accept 10 papers (acceptance rate 43%) including 7 full papers, two short papers, and one experience report. Furthermore, we accepted the two submitted industry talks and invited 4 more industry talks for having a balanced program between research and industry contributions. Authors from accepted papers came mainly from Europe (50%), Asia (25%), Oceania (19%), and the Americas (6%).

ICSSP 2023 had about 30 participants on-site and additional people participating online (especially steering committee members joining for the opening and closing session). Registered participants of the conference came mainly from Europe (40%), Asia (30%), Oceania (27%), and the Americas (3%).

Overall, there was a lively and stimulating discussion after presentations and during the breaks how emerging technologies have an impact on software and system development processes.

4 Program Summary

The program of ICSSP 2023 comprised three keynotes, two full paper sessions, one short paper and experience report session, one industry talk session, and one industry panel discussion session.

The first keynote talk "A.I.gile - How Agile and AI (e)merge in practice" was given by Prof. Dr. Philipp Diebold of IU International University and Bagilstein in Germany. His presentation focused on how Artificial Intelligence (AI) can be integrated into Agile processes. It explored the ways in which AI can enhance Agile methodologies and improve the delivery of software solutions. The presentation discussed the challenges and considerations that organizations must consider when implementing AI in Agile environments. Attendees gained insights on how AI can improve Agile processes and deliver innovative and valuable solutions.

The second keynote talk "A Technical Focus on Business Process Management – Past, Present, and Emerging Topics" was given by Prof. Dr. Ingo Weber TU Munich and Fraunhofer Gesellschaft in Germany. The goal of this talk was to provide an overview of the Business Process Modelling (BPM) field, which is very related to software engineering in several ways, and responsible for two out of nine German unicorn startups. Accordingly, the talk started with a short excursion into history, including a 100-year-old process model. Then he discussed process enactment and its relation to low and no-code development and model-driven engineering. The past

10 years of BPM were heavily influenced by process mining, a flavor of data mining using a process lens. The talk gave several examples using emerging technologies like blockchain and finished with a brief view on augmenting BPM technologies with artificial intelligence (AI).

The third keynote talk “AI-Augmented Software Engineering: Revolutionizing or Challenging Software Quality and Testing?” was given by Dr. Tafline Ramos, Amanda Dean, and David McGregor of Planit, an NRI Company. ChatGPT itself offers great optimizations to initial research and document writing for industry practitioners. However, it also suffers from key issues, such as not being able to reference the sources of its ideas. It also has the potential for making incorrect inferences that could be mistakenly accepted as factual by end-users, used in workplace documentation, or in the worst case, published on the internet, and then utilized in future releases of AI-based tools, creating negative feedback loops in human learning. The talk looked at the current benefits and shortcomings of AI-based software quality improvement tools, and the challenges of testing AI-based software from a Quality Engineer’s perspective. They looked at international standards and training courses for AI development and testing. They also explored problems that quality engineering practitioners would like to see solved in the future, to enable a holistic approach to AI-augmented software quality engineering.

Full paper session one was about “Process Automation” and had three paper presentations: (1) “Automatically Generating Docker Files via Deep-Learning: Challenges and Promises” by Giovanni Rosa (University of Molise), Antonio Mastropaolo (Università della Svizzera italiana), Simone Scalabrino (University of Molise), Gabriele Bavota (Software Institute, USI Università della Svizzera italiana), and Rocco Oliveto (University of Molise). (2) “An Experience Report on Assessing Software Engineer’s Outputs in Practice” by Juzheng Zhang, He Zhang, Lanxin Yang, Liming Dong, and Yue Li (all (Nanjing University). (3) “Automatic Detection of Security Deficiencies and Refactoring Advises for Microservices” by Burak Ünver and Ricardo Britto (Ericsson and Blekinge Institute of Technology).

Full paper session two was about “Process Improvement and had four presentations: (1) “Using GUI Test Videos to Obtain Stakeholders’ Feedback” by Jianwei Shi, Jonas Mönnich, Jil Klünder, and Kurt Schneider (all Leibniz Universität Hannover). (2) “Measuring the Benefits of CI/CD Practices for Database Application Development” by Jasmin Fluri (Schaltstelle GmbH), Fabrizio Fornari (University of Camerino), and Ela Pustulka (FHNW). (3) “On Preparing and Assessing Data for Simulation Process Modeling: An Industrial Report” by Liming Dong, He Zhang, Yue Li, Bohan Liu, and Zhiluo Weng (all Nanjing University). (4) “Adding Generic Role- and Process-based Behaviors to Smart Contracts using Dynamic Condition Response Graphs” by Yibin, Tijs Slaats, Boris Düdder, and Thomas T. Hildebrandt (all University of Copenhagen).

The industry talks session had six presentations: (1) “De-risking major industry transformations using experimentation” by Gayan Benedict (University of Technology Sydney). (2) “Analyzing the Implications of NFTs as Digital Identities in Web3.0: A Privacy-Focused Approach” by Memoona Anwar, Tony Fitzgibbon and Inam Gull (Data Zoo). (3) “Considering the Human in the loop when adopting AI” by WHOM?. (4) “Helping companies de-risk the integration of first, second and third-party AI models into a customer engagement platform” by Justin Tauber (Salesforce). (5) “Achieving Business Success - A Framework for Software Delivery of Emerging Technologies” by Mahesh Venkataraman and Shylaja Shivaram (Accenture). (6) “Improve Software and System Process of Cloud Serverless Architectures through Automated Testing with AI” by Rohit Patwardhan and Mallika Fernandes (Accenture).

Presenters of our industry talks also participated in the panel discussion talking about challenges and solution approaches of emerging technologies in the context of software and system development processes.

In the final session two short papers and one experience report were presented: (1) “Characterizing the Impact of Culture on Agile Methods: The MoCA Model” by Michael Neumann (University of Applied Sciences & Arts Hannover), Klaus Schmid (Stiftung University Hildesheim), and Lars Baumann. (2) “Towards Sustainable Software for Public Sector Information Systems” by Reetta-Kaisa Ghezzi, Aapo Koski, Janne Lautanala, Mikko Lehtisalo, Manu Setälä, and Tommi Mikkonen (University of Jyvaskyla). (3) “Towards Better Code Reviews: Using Mutation Testing to Improve Reviewer Attention” by Ziya Mukhtarov, Mannan Abdul, Mokhlaroyim Raupova, Javid Baghirov, Osama Tanveer, Haluk Altunel, and Eray Tüzün (all Bilkent University).

5 Distinctions

Based on review scores of PC members, we decided to have two best paper awards, which were handed over during the closing session.

The first award went to the short paper “Towards Better Code Reviews: Using Mutation Testing to Improve Reviewer Attention” by Ziya Mukhtarov, Mannan Abdul, Mokhlaroyim Raupova, Javid Baghirov, Osama Tanveer, Haluk Altunel, and Eray Tüzün (all Bilkent University).

The second award went to the full paper “Using GUI Test Videos to Obtain Stakeholders’ Feedback” by Jianwei Shi, Jonas Mönnich, Jil Klünder, and Kurt Schneider (all Leibniz Universität Hannover).

6 Future

ICSSP 2024 is planned to be co-located with the 46th ICSE in Lisbon, Portugal. We look forward to another inspiring 18th edition of the ICSSP conference.

ACKNOWLEDGMENTS

We want to thank our devoted members of the organizing and program committees for their support, allowing us to arrange and run an exciting conference and to assemble a high-quality program. We also would like to thank the steering committee for the continuous support. Finally, we express our sincere gratitude to all authors who shared their research insights and practical experience with us as well as the presenters who spark stimulating discussions with all the participants, who made ICSSP 2023 a special and great conference.

COSMIC News

summarized from the COSMIC International Advisory Council (IAC) members meeting by Reiner R. Dumke

COSMIC Certification 2022 – 2023

Jean-Marc Desharnais

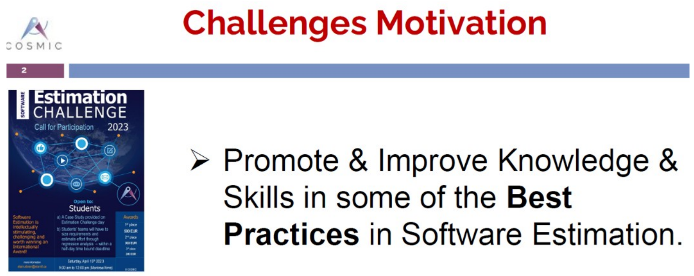
Country	Number
Canada	24
China	80
Grèce	2
Maroc	1
Mexico	20
Netherlands	1
Poland	15
Total	143

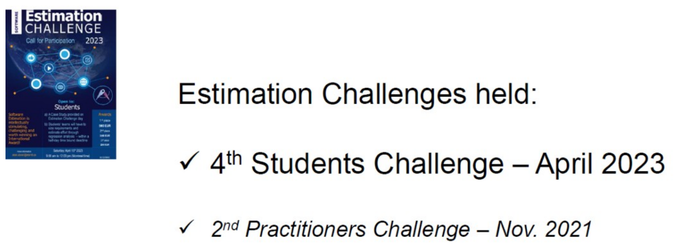
CONCLUSION

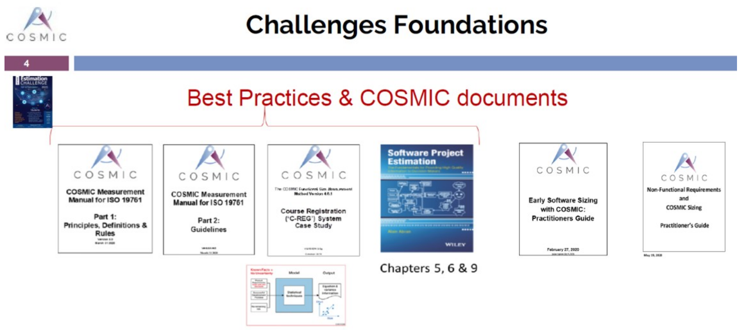
- 78% of participants passed their certification exam.
- This year China is the most represented when it comes to the number of registrations and certifications
- Two of the three organizations with the most certifications are Chinese
- Two new countries has been added to the registration requests: Costa Rica and Morocco
- The certification for the size estimate did not have a single request

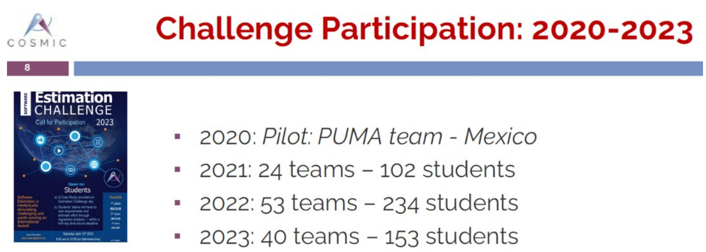
Estimation Challenge committee Annual Report 2022-2023

Alain Abran









COSMIC Awards 2023

K. R. Jayakumar

5 year History of Awards

- COSMIC Awards Launched in 2018
- This is 6th year of COSMIC Awards
- Earlier Awards
  - IWSM 2018, China, - Special Life Time Achievement Award
  - IWSM 2019, Haalem, Netherlands – Research Award, Certifications Award & Benchmark Awards
  - IWSM 2020, Mexico – Certifications Award & Benchmark Award
  - IWSM 2022, Turkey – Certifications Award

COSMIC

3

Life Time Achievement Award

Presented to

Charles Symons

2018 September 19, IWSM 2018, Beijing, China

Research Award 2019

Certifications Award 2019

Benchmark Award 2019

Certification Award Winner 2020

Benchmark Award Winner 2020

Certification Award Winner 2022

Certification Award Winner 2023

GUFPI-ISMA News

Luigi Buglione

ISMA President, Italy, luigi.buglione@gufpi-isma.org

Upcoming events from GUFPI-ISMA:

- Oct. 29, 2023: Webinar on “ISO 16355: the history of Quality Function Deployment (QFD)” with Thomas Fehlmann
  https://gufpiisma.wildapricot.org/event-5419180

- Nov. 24, 2023: 2[d] Metric Event 2023 (Bari/Italy)
  https://gufpiisma.wildapricot.org/event5419026

All the previous events are available at: https://gufpiisma.wildapricot.org/events

BWT about ISBSG:

- IT Conference 2023 (online), Nov. 10, Nov. 17, 2023
  https://www.isbsg.org/it-conference-2023

Resolving the historical confusions about the meaning of software size and its use for project effort estimation

Charles Symons

(Retired, unaffiliated) Reigate, England. E-mail: cr.symons@btinternet.com

This version of this paper has been updated in a number of places from the version submitted to the IWSM Mensura Conference in Rome and presented on 15th September 2023

Abstract.

The software industry does not have a good track record of delivering systems on time and budget. In part this is due to weaknesses in software sizing and project effort estimating methods.
This paper discusses how some of these weaknesses have arisen historically, resulting in differing views on some basic underlying concepts such as the meaning of software size, types of software size and their relationships, the distinction between software 'size-drivers' and project 'effort-drivers', the meaning of the weights assigned to the components of software functional size measures, and the meanings of Functional User Requirements and of Non-Functional Requirements. Several conclusions are drawn, particularly that the current definitions of the latter two concepts are misleading in how they deal with quality requirements. Some recommendations are made on the need for 'coherent size/effort ecosystems' to help improve project effort estimation, and to revise some existing concept definitions.

Keywords:

Software measurement, project effort estimation, functional requirements, non-functional requirements

1. Introduction

75 years after the term 'software engineering' was first coined, there is still no common understanding of what we mean by the 'size' of a software item, and where to draw the line between methods for measuring software size, and methods for estimating the effort for a project to develop an item of software starting from an estimate of the software's size.

This paper aims to examine the origins from a historical viewpoint of some of the confusions that are common to all these methods where there is evidence that the confusions are still prevalent today, and to suggest some remedies to help eliminate the confusions.

Having been actively involved in many of the developments since Allan Albrecht first published his ‘Function Point’ (FP) method for sizing software requirements [1], and at last believing I can explain the origin of these confusions, one of my aims is to clear my own conscience. In what follows, where I criticise a development, I confess to being directly or indirectly involved in contributing to the historical confusion. Please forgive me. Along the way, I reach some new (to me!) conclusions that I wish I had realised years ago.

In writing this paper, I have assumed that the reader is familiar with the two most important uses of software sizes, namely

a) for measuring the performance of a completed software project (e.g.

project productivity = work output/work input = software size/project effort),

and

b) in ‘top-down’ methods for estimating the effort for a project to develop a new item of software.

These effort-estimation methods require as input: an estimate of software size (usually the largest driver of effort); other system or project requirements; and past (or ‘benchmark’) performance data.

I also assume that the reader is familiar with common software size measures such as counts of Source Lines of Code (SLOC) and is familiar, at least in outline, with the most widely known Functional Size Measurement (FSM) methods, i.e. IFPUG [2], Nesma [3], MkII [4] and COSMIC [5], and with some effort estimation methods. All these methods have been analysed and criticised at length for their strengths and weaknesses and their suitability for their claimed purposes. Again, I assume the reader is aware of these discussions.

The methods of software sizing listed above have become less commonly used in recent years under multiple influences such as the increasing practice of assembling software from existing components and from the introduction of Agile methods. The latter, when first introduced, added to our list of confusions by using Story Points [6] to size short statements of software requirements. In practice a size estimate in units of Story Points was typically also taken as an effort estimate. e.g. by assuming that

one SP = one work-day.

The result was that a measurement of ‘velocity’ (a more developer-friendly term than ‘productivity’), i.e. estimated SP/work-day, was actually a measure of estimation accuracy rather than of performance. Nowadays, it seems to be acknowledged that use of Story Points is a ‘bottom-up’ means of effort estimation.

In spite of their less common use nowadays, methods for software sizing and project effort estimation, are still highly relevant to cases where a customer requires an estimate of software development project cost in order to make the business case for the investment, or to reach an agreement on costs with a software supplier. This is especially the case for the development of very large software systems. Software sizing methods are also important for quality control on software requirements, for managing project 'scope creep', for monitoring performance-improvement initiatives, and the like. In my opinion, it is therefore still important for future generations of software engineers to resolve some of the confusions that still persist today in this subject area.

2. So what do we mean by the 'size' of a software item?

Answering this question is important if we are to agree what parameters to include in our software size measures.

When Allan Albrecht first described his FP method [1], he defined his measurement as giving '*an effective relative measure of function value delivered to our customer'*. This statement initially caused much confusion. It was a nice marketing description, but the size of an item of software does not necessarily have any relation at all to its value[1] to the customer.

Software size is a rather nebulous concept that is difficult to define. In ordinary discourse, we think of the size of a software item as a measure of *how big it is*, or as a measure of the *amount* of software *product*, i.e. the *work-output*, of a project. In ordinary conversation, we often talk about 'the' size of an item of software, only distinguishing the results of using different methods of measuring 'the' size. There is no one thing that one can call as *the* size of an item of software.

An item of software evolves over its life across different states, from an expression of requirements to an executing program. Figure 1 shows the life-cycle of a software item (the 'Entity') developed following a 'waterfall' approach, the artefacts[2] that exist in each state (the Entity's Attributes) and the Measures of a size of those attributes, using the EAM taxonomy [7].

Therefore a measurement of software size can only be properly understood when the following are all known: the state of the item's artefacts being measured, the specific method used for the measurement, and the unit of measurement. Some brief observations on Figure 1:

----

[1] Software certainly delivered value in the form of revenue to IBM (Albrecht's employer), likely in proportion to its size. But the value to the customer could range from negative if the software was never used, to many times its cost if it helped the customer achieve business efficiencies or increase revenue.

[2] These artefacts have their own life-cycle. For example outline requirements may be measured approximately by one method then, when the requirements are known in more detail, re-measured more precisely by another method.

- 'Functional size' is generally accepted as meaning a measure of the size of 'what the software must do'. For a fuller discussion of Functional User Requirements and functional size, see section 4.
- Figure 1 shows a question-mark against sizing 'Non-Functional' Requirements (NFR) because there is confusion on how to distinguish functional from non-functional requirements and how to measure the latter, if at all. This topic is discussed in section 5.)
- Object Points are defined in Wikipedia as '… *an approach used in software development effort estimation* …. (they) *are a way of estimating effort size, similar to Source Lines Of* or *Function Points*.' (This is another perfect example of confusing software size and project effort.)

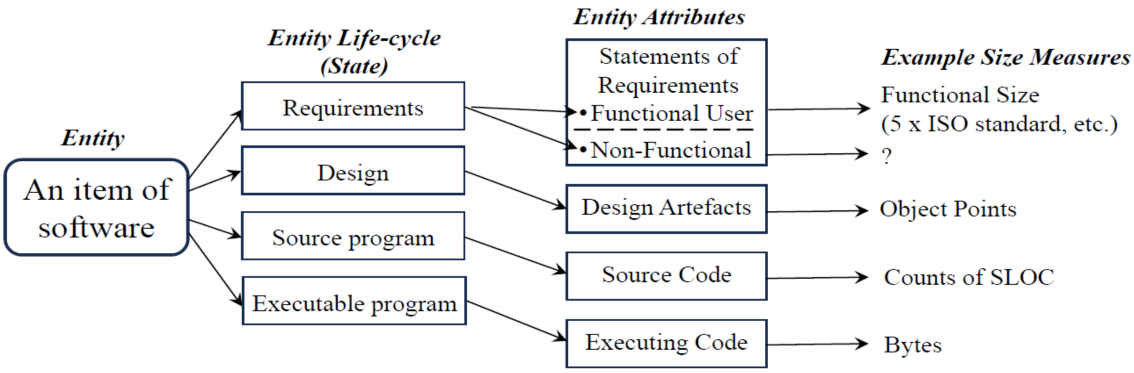


Figure 1. A software item (the Entity), its Attributes and size Measures over its life.

When we discuss a physical object, we usually talk of its size in terms of the measure of its major dimension, for example the height of an office block, or the tonnage of a ship. The nearest dimension of a physical object analogous to a measure of software size is its *length*. Examples of useful measures of the length of a software item include a count of the SLOC or of the number of bytes occupied in computer memory. The length of a software item may not, however, be an adequate measure of the *difficulty* of developing it.

Given the use of software size measures for effort estimation, an underlying aim of every designer of a software size measurement method has been that, for a given homogeneous set of projects, the method should measure software sizes that correlate reasonably with the effort of the projects to develop those sizes. A size measure that cannot be demonstrated to be reasonably correlated with effort is practically useless. Some software size measurement methods have therefore tried to account for other factors than length to reflect this extra difficulty, such as the 'complexity' of the software (a complex subject!) .

After Albrecht first described his method, the IFPUG organization assumed responsibility for the method and refined its definition. A size in units of FP [2] was defined as the product of two numbers: the 'Unadjusted Function Points' (UFP) size, and the 'Value Adjustment Factor' (VAF) size.

Measuring a UFP size involves first identifying the occurrences of five 'Base Functional Component' (BFC[3]) types in the item's functional requirements. Next, each of the BFC types are classified into one of three sizes depending on their 'complexity', where the latter depends on two dimensions (e.g. counts of Data Element Types (DETs) and of File Type References (FTRs)). The 5 x 3 = 15 different possible BFC-type sizes are finally allocated a fixed number of FPs on a common, *one-dimensional* size scale. At this point, the foregoing two-dimensional classification parameters are 'forgotten'. This procedure is fundamentally no different from counting SLOC as a measure of the size of a software program. One SLOC BFC can be of several types (declarative, control, compute, etc.) and has a length in terms of the number of characters in the line which can vary enormously. Nevertheless, we ignore this detail and count each SLOC as one unit.

Given that we are accustomed to regarding a SLOC count as a measure of 'source code length', by the same logic we can regard a count of UFP as a measure of 'functional length', and a count of OO Points as a 'design length'.

IFPUG's VAF size, however is of a different nature. It accounted for 14 factors, mainly of various technical requirements that were judged to affect software size apart from its length. For example, in the early 1980's more work had to be done, i.e. it was more *difficult*, to develop a system to operate on-line than in batch mode. It therefore made sense to include various types of requirements in the FP size measure that accounted for such extra difficulties. One can criticise the way the VAF was designed, but Albrecht's intention in first designing the VAF was, in my opinion, perfectly understandable. I will refer to these other factors in his software size measure that reflect the goal of taking account of development *difficulty*, as 'size-drivers'.

Conclusion.

There is no reason to expect that different methods for measuring the 'length' of software items at different states in their life-cycle will produce results that correlate well with each other, or with measurements of the same items that attempt to also account for the difficulty in developing the sizes, or that account for this difficulty in different ways.

Yet many results have been published on the degree of convertibility between sizes of software in different states, measured by different methods, without acknowledging these factors or attempting to compensate for the inevitable differences that arise.

----

[3] A BFC is defined as an 'elementary unit of Functional User Requirements defined by and used by an FSM Method for measurement purposes'. [11]

Although functional size is usually the main driver of the effort of a project to develop a new item of software, there are many other types of requirements for a software project besides functional size that must be taken into account when estimating the project effort. I will refer to the latter as 'effort-drivers'. Confusion now arises because designers of different sizing and estimation methods have made different decisions about which of these requirements to consider as software size-drivers and which as project effort-drivers.[4]

Listed below are a few of the main groups of requirements that, historically, have been classified as *either* software size-drivers *or* as project effort-drivers[4].

a) Quality requirements such as for performance, reliability, etc.
b) The complexity of the software, particularly of mathematical algorithms and logic sequences.
c) The programming language used to develop the software
d) The number of users and/or implementations.
e) The extent of the software product that will re-use existing software components.

Some requirements are, however, 'true' project effort-drivers, meaning they are attributes only of the project to be developed. For example:

a) Project processes, risk, and governance.
b) Project constraints such as target delivery dates, budget limitations, inter-dependencies with other projects, etc.
c) The project staffing, taking account of the actual staff numbers available and their experience relative to the ideal needs.

Figure 2 illustrates the choices to be made for any process of estimating project effort starting from an estimate of software size (in this case assuming functional requirements as input), showing the other requirements that may be allocated as either software size-drivers or as project effort-drivers (from the first list above), and the 'true' project effort-drivers (from the second list).

----

[4] This paper focuses on the confusions arising in sizing and effort-estimation for a project to develop a software-item. The actual *processes* of a) estimating effort from the various types of requirements/effort-drivers, taking into account measures of past performance in delivering systems of the type to be developed (i.e. benchmarks) and then b) for converting estimated effort and other requirements into project *costs* are beyond the scope of this paper. This also explains why I avoid using the term 'cost-driver'.

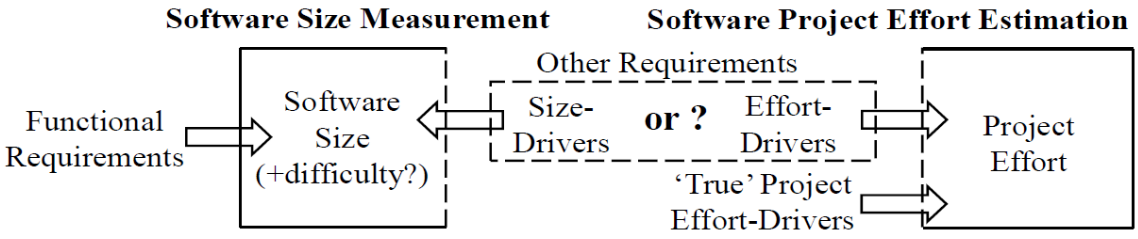


Figure 2. Which requirements to consider as 'Size-Drivers' and which as 'Effort-Drivers'?

Recommendation.

Any method that aims to make a reasonably accurate estimate of project effort must ensure a coherent and consistent allocation of size- and effort-drivers between the software sizing step and the effort estimation step.

Let us call the resulting system a **'Coherent Size/Effort Ecosystem'**.

As examples:

- The COCOMO estimation method [8] which takes counts of lines of code as the software size measure is a coherent eco-system (leaving aside whether the system has other possible limitations or deficiencies).
- The commonly-used procedure of estimating the size of the software functional requirements, then converting this size to counts of lines of code using external data, then entering this size into a black-box effort estimation tool, is certainly NOT a coherent size/effort eco-system and is most unlikely to produce consistently reliable effort estimates. Santillo [9] pointed out how easily errors can propagate when using such a process.
- To establish a coherent *in-house* eco-system for estimating effort early in the life of a project, probably the best way (if one has the resources) is to use a standard method for sizing software functional requirements, to define a limited set of effort-drivers relevant to the in-house environment, and to collect sufficient size and effort data on past completed projects to establish in-house performance benchmarks.

3. On the 'weights' used by some early methods for sizing software requirements.

When I first used Albrecht's sizing method, I found some systems where the measurements did not seem to fully reflect the size of transactions that had to navigate through large, multi-level (i.e. 'complex') databases. My main aim in developing the 'MkII' FP Analysis method [4], therefore, was to improve on Albrecht's Unadjusted FP size component.

I chose to measure the size of a logical transaction by a count of the DETs as the measure of each of the input and output phases of the transaction respectively, and by a count of the number of entity-types referenced (ERs) as the measure of the processing phase of the transaction. But how to add two counts of DETs to a count of ERs? The three counts had to be weighted in some way. It seemed obvious to use weights proportional to the relative effort to implement the three types of counts. I therefore used a Delphi approach, asking the developers of over 60 projects to 'guestimate' the relative amount of effort needed for the three phases of the transactions. From these data I derived an 'industry-average' set of weights. The MkII UFP size of a transaction was then the weighted sum of the counts of the DET's and ERs.

At the time, I saw nothing inherently wrong with Albrecht's VAF, so I added a few more factors, e.g. requirements for security, interfaces to other applications, etc., re-named the result a 'Technical Complexity Adjustment' (TCA), and re-calibrated the weight of the sum of its components, using the same Delphi approach. Later, I received evidence[5] that Albrecht's weights for the components of his FP method were similarly derived from an IBM effort estimation method, i.e. Albrecht's weights were also derived from relative effort to develop the various components.

Shortly after publishing the MkII method, a software metrics expert for whom I have great respect, commented that MkII FPA was '*not a software sizing method, but an estimation method'*. So, is it true that these early Function Point methods, [2], [3], [4] and others are actually 'software sizing' methods? Or are they 'project estimation' methods?

Undeniably, from a mathematical viewpoint, given the weights applied to counts of BFCs were all derived from relative effort, the units of the methods must be proportional to effort. But the *Unadjusted* FP sizes they produce only take account of requirements for software functional 'length'; they do not take account of any of the other factors that may be considered as size-drivers or effort-drivers of the project being estimated, So these methods cannot, in my opinion, be considered as estimation methods in any practical sense; they are hybrids of software sizing and effort estimation methods.

Conclusion.

The UFP components (BFCs) of these early FP methods actually measure a standard 'Relative Effort' for a software project.

Describing these methods as measuring a standard 'Relative Effort' is more accurate than describing them as a standard 'functional length'. A Relative Effort size is a valid measure (or an 'index' if preferred) of the amount of work required to develop the functionality of an item of software, relative to an arbitrary standard work-size. Relative Effort size measurements are on a ratio scale. Their units of measurement have no meaning on any absolute scale (like a 'Dow Jones index' of software size, as Albrecht once commented.)

----

[5] I was given a paper copy of a set of Albrecht's hand-drawn OHP slides entitled "Where Function Points (and weights) came from", dated February 2nd, 1986

The concept of defining a 'standard effort' as a measure of the size of a task, against which actual effort could be compared was first introduced by Frederick Taylor [10]. His ideas on measuring work and on using them to measure and help improve productivity have been in use for over a century.

Taylor's ideas are applied to highly-repetitive work and his standard effort is measured in absolute units of time (e.g. minutes) for a specific process. In contrast, software development is non-repetitive work and Relative Effort is a measure that is independent of the development process. Nevertheless, the idea of comparing actual effort against a measure of standard effort in order to measure productivity is the same for both cases, and equally valuable. Similarly, the estimated Relative Effort of a software-item to be developed may be used as the primary software size input to a project effort estimation method.

Does it then follow that a Relative Effort size is also a valid measure of the *functional size* of a software-item, as the methods' protagonists claim?

4. The ISO/IEC standard 14143/1 on 'Functional Size Measurement: Definition of Concepts'

In about 1995, the International Organization for Standardization (ISO) established a Working Group (ISO/IEC/JTC1/ SC7/WG12). WG12 set out to define some principles for Functional Size Measurement. The resulting standard, ISO/IEC 14143/1 [11], includes some important definitions.

> "**Functional User Requirements** (abbreviated as 'FUR'): *sub-set of the User Requirements describing what the software does, in terms of tasks and services*."
>
> "**Functional Size**: *size of the software derived by quantifying the Functional User Requirements.*"

[A first, maybe pedantic, comment about the definition of 'FUR' is to ask why are FUR a sub-set of the *User* requirements? Requirements may be specified by many actors, including the project sponsor and lawyers, who will never be users. What the term intends, I believe, is that these are the requirements for tasks and services that will be *provided to* the software users. Secondly, the term 'functional' (or function, or functionality) is not defined in 14143/1. The meaning is only implied by the phrase '*what the software does'*. These two interpretations of the term FUR turn out to be quite important - see Section 5. Another minor anomaly in the definition of FUR is that in reality Requirements specify what the software 'must or should do' *in the future*, when it is developed – not what the software 'does', *implying it already exists*.]

Recommendation.

A better definition of FUR would be 'sub-set of the requirements describing what the software must or should do, in terms of tasks and services, for its users.

One of the most important FSM principles defined in 14143/1 is (extracts):

"Functional Size shall have the following characteristics:

i it is not derived from the effort required to develop or to support the software being measured;
ii it is independent of the methods used to develop or to support the software being measured;
iii it is independent of the physical or technological components of the software being measured."

Conclusion.

The two definitions quoted above and the clause i) concerning FSM characteristics mean that ***the 'Unadjusted' FP size components of early Function Point methods that rely on relative-effort-related weights (i.e. that measure Relative Effort sizes) can legitimately describe themselves as 'FSM Methods'.***

It follows, however, that the VAF component of Albrecht's method, the TCA component of the MkII FP method and their equivalents in other early FP methods do not comply with clauses ii) and iii) of the FSM characteristics. These components were therefore dropped for ISO FSM method standardization purposes and were, effectively, consigned to history.

5. Functional or Non-Functional Requirements?

The VAF and their equivalents in other FP methods had fulfilled a role as software size-drivers in the then-existing size/effort eco-systems (coherent or not). Now that their components no longer contributed to functional size, they inevitably had to join the list of effort-drivers and so they became entangled with the concept of 'Non-Functional Requirements'.

A web-search on 'Non-Functional Requirements' (NFR) reveals a plethora of different lists of example NFR and a variety of vague definitions, many published in recent years. How to define NFR, how to distinguish NFR from FUR, and whether it makes sense to measure a size of a set of NFR (as indicated in Figure 1), are still major sources of confusion in the software metrics community.

For many years the IEEE's SEVOCAB [12] gave a definition of NFR from the ISO/IEC/IEEE 24765:2010 standard as:

"A software requirement that describes not what the software will do but how the software will do it. Example: software performance requirements, software external interface requirements, software design constraints, and software quality attributes."

The 14143/1 definition of FUR has a (non-normative) Note, part of which effectively provides us with another definition of NFR. It states:

"User Requirements that are not Functional User Requirements include but are not limited to:

i. quality constraints (for example usability, reliability, efficiency and portability);
ii. organizational constraints (for example locations for operation, target hardware and compliance to standards);
iii. environmental constraints (for example interoperability, security, privacy and safety);
iv. implementation constraints (for example development language, delivery schedule)."

Notice the overlaps (but also the inconsistencies) between the components of a VAF or TCA (described earlier as 'size-drivers'), the examples of NFR given in the SEVOCAB definition, and the examples of requirements that are *not* FUR according to ISO/IEC 14143/1.

The early definition given in SEVOCAB is not very helpful. None of the examples it gives of NFR define 'how' the software will do what it must do. Moreover, some performance requirements typically apply at the *system* level e.g. requirements for response time or availability; these can therefore involve requirements for hardware as well as software.

Recognising the lack of a precise definition of NFR, the COSMIC and IFPUG organizations collaborated to produce a more refined definition of NFR and a glossary of 70 NFR terms [13]. Their definition of NFR[6], which now also appears in the SEVOCAB, is as follows.

"A*ny requirement for a software-intensive system or for a software product, including how it should be developed and maintained, and how it should perform in operation, except any functional user requirement for the software.*

NOTE: Non-functional requirements (NFR) concern:

- *the software system or software product quality;*
- *the environment in which the software system or software product must be implemented and which it must serve;*
- *the processes and technology to be used to develop and maintain the software system or software product, and the technology to be used for its execution."*

----

6 (which I personally initially drafted!)

Taken together, the 14143/1 definition of FUR and this latest definition of NFR were intended to encompass all the possible requirements for a software-intensive system or software product, and to be mutually exclusive. In other words, any software system requirement must be *either* a FUR *or* a NFR. But with hindsight this is not true. In fact, making this hard distinction between FUR and NFR turns out to be highly misleading and has added to our confusions.

The problem arises with classifying *quality* requirements as non-functional according to the definitions of both FUR and of NFR. But to take a simple example, a quality requirement for security may be implemented either in software (hence arise from a FUR), or in hardware (hence arise from a NFR), or by a mixture of both software and hardware.

More importantly, many quality requirements may initially be expressed as NFR, but are likely implemented *entirely* in software and thus contribute to the size of the software. Examples are requirements for auditability, privacy, portability, maintainability, usability, etc. Such requirements could therefore equally be first expressed in statements of NFR or in statements of FUR.

Any requirement, whether initially expressed as a FUR or as a NFR, that is allocated to software must add to its size, which will show up in the software's size when measured by e.g. counts of SLOC or of bytes. However, FSM methods generally cannot account for every type of software requirement in their measures of functional size – requirements for mathematical algorithms being an obvious example. This inability of FSM methods to account for all types of requirements can lead to difficulties in their use as the primary input for estimating effort.

An example of the current confusion surrounding measurement of NFR exists in the form of the SNAP method [14]. The impact on software size and/or project effort of individual NFR can of course be measured or estimated. But the SNAP method aims to define how to measure a 'non-functional size' for software which in turn, it claims[7], enables one to measure a standard *collective* size of the NFR for a software item. (The method assumes that NFR are defined as above by COSMIC/IFPUG but are limited to NFR for software). However, in my opinion, regardless of the validity of this claim, it was unwise to attempt to define a measure of the collective size of any set of NFR for a software item for several reasons.

----

[7] The SNAP ('Software Non-Functional Assessment Process') method was apparently designed to measure a size of various types of requirements for software features that could not be accounted for by the IFPUG Function Point method. The size resulting from a SNAP measurement was called the 'Non-Functional Size' to distinguish it from the Functional Size. This decision was unwise: classifying a software feature as 'non-functional' simply because it cannot be measured by a FSM method is confusing. Next, 'requirements for the non-functional size of software' were taken to mean the same as 'the non-functional requirements for software', an entirely different concept. The claim that the SNAP method measures a size of the 'NFR for software' is therefore disputed [15].

- It is extremely difficult, if not impossible, to envisage a *meaningful* collective size of such a wide variety of *types* of NFR for software, which range across requirements for software quality, the environment it must serve (e.g. including the size of the user base), and the technology to be used for its development (e.g. including the programming language to be used, re-use of existing code, etc.)

- Not only is there a wide variety of *types* of NFR for software, but also *numbers* of possible NFR. For example, the COSMIC/IFPUG Glossary lists around 30 terms for different quality requirements for software, some of which overlap in meaning. This is no basis for a standard size measurement.

- The method defines 14 different BFCs, each of which can have 3 sizes. Applying expert-judgement to determine the 42 effort-related industry-standard weights for the various contributions is a challenge.

- As we have seen, quality requirements may be viewed as NFR in some cases and by others as FUR. (Indeed, at the time of writing, the IEEE/ISO/IEC Committee Draft version of the SNAP standard [14] acknowledges that '*The boundary between functional requirements and NFR does not have a universally-agreed definition'* and *'This document covers a subset of non-functional requirements'* - but does not state which sub-set. Again, this is no basis for a standard size measurement.

- It must be a serious challenge to design a coherent size/effort eco-system when having to incorporate both a functional size and a NFR size on different unrelated scales, to estimate the allocation of effort between work on the two sizes, then to work out how the functional and NFR sizes relate to existing size- and effort-drivers, and to existing benchmark data (probably based only on functional size).

Conclusion.

We must recognise that any requirement for a software system project, however initially expressed, must ultimately be allocated

- ***either to software functionality, and thus contribute to its size (though only counts of SLOC or bytes may be able to detect every size addition),***

- ***or to project effort or other project costs. Such requirements can arise from non-software-related items such as for hardware, or for other activities which may consume project effort such as hardware installation or training, or the requirement must be a 'true' project effort-driver, i.e. a constraint on the project.***

In this context, the current ISO/IEC definition of FUR and the COSMIC/IFPUG definition of NFR create confusion because they classify quality requirements as non-functional, but we know that quality requirements can impact software and/or non-software. To resolve this issue and hence the confusion about the distinction between FUR and NFR requires changes to their respective definitions.

Recommendation.

Remove clause i) in the NOTE to the definition of FUR in the ISO/IEC 14143/1 standard concerning 'quality constraints' and remove the corresponding Note in the 'COSMIC/IFPUG Glossary of terms for NFR (etc)' concerning 'the software system or software quality constraints'.

In both of these documents, replace the deleted clause by: 'NOTE. Quality constraints or requirements may be expressed as either functional or as non-functional requirements.'

A more radical, alternative option is to re-think the concept of NFR with a new definition that better reflects the name 'NFR':

Alternative Recommendation.

'Any requirement for a software-intensive system or for a software product that does not add to software functionality.

NOTE: Non-functional requirements concern organizational constraints (for example, numbers of implementations), the processes and technology used to develop and maintain the software system or software product, and the technical environment in which it is executed.'

In practice a consequence of the existing definition is that whereas quality requirements for, say, 'system availability' or for 'system performance' will be interpreted initially as a NFR, in the future, with either of the proposed options for the new definitions, it will not be clear initially whether such a requirement implies a FUR or is an NFR, or a mixture. If that forces earlier consideration of this question, that may be an advantage of the new definition.

Accepting the above Conclusion and the second Alternative Recommendation should mean that the model of Figure 2 can be rationalized making it easier to develop a Coherent Size/Effort Ecosystem, as shown in Figure 3.

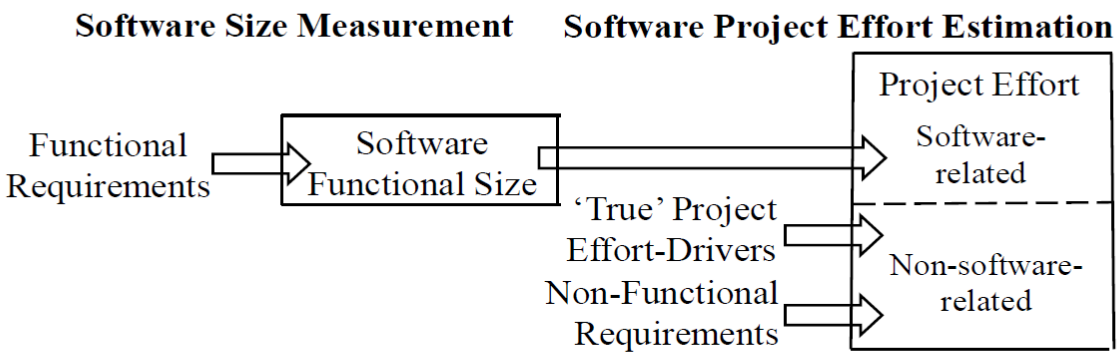


Figure 3. A simplified Coherent Size/Effort Ecosystem, assuming the recommended alternative revised definition of NFR

6. Overall Conclusions

The community of professionals interested in software size measurement and project effort estimation methods is relatively small and fragmented. This has perhaps led to the current inconsistent set of concepts and definitions. In turn, this has resulted in practitioners having to work with *incoherent* size/effort ecosystems, often producing poor effort estimates. The software industry's poor reputation for delivering systems to estimated time and budget seems likely to be due in part to the weaknesses of its software sizing and project effort estimation methods and products. These weaknesses then feed back into their limited use. A vicious spiral.

The various methods for measuring a size of a software item discussed in this paper all produce one-dimensional size measures, but they have different meanings with consequences for how they can be used to build coherent size/effort ecosystems.

- Sizing methods, the weights of whose BFCs were calibrated on project effort, with units expressed as 'Unadjusted' Points (e.g. early UFP sizes, Unadjusted Use Case Points, and the like) are really measures of the amount of work required to develop the functionality of a software item, relative to an arbitrary standard amount of work.
- The early UFP sizes qualify as measures of a 'functional size' of the FUR, according to the ISO/IEC 14143/1 definition. They can also be thought of as measures of 'functional length'.
- The COSMIC FP method produces functional sizes that are truly independent of effort (but that have been shown to correlate well with effort for several types of software at different levels of granularity). CFP sizes, with a single 'data movement' as a unit of measurement can also be thought of as a measure of 'functional length'.
- A count of the number of bytes of memory that a program occupies when it is executing is the software size measure that comes closest to our notion of a physical length.
- One line of SLOC can be thought of as a measure of 'source code length' but SLOC counts suffer weaknesses as a standard due to varying counting rules, dependence on programming language, programmer skill, and other factors.

As far as measurement practices are concerned, two priorities stand out for implementation from this discussion.

For the software sizing and estimation community generally, the greatest needs are to converge on a common understanding of the meaning of and relationships between FUR and NFR, and to define coherent size/effort eco-systems. This should help improve understanding and acceptance of the subject of software size measurement and improvement of effort estimation methods.

If the Agile community is serious about measuring its productivity and demonstrating improvement, then it needs to adopt an objective, standard measure of its work-output and to build a coherent 'top-down' size/effort ecosystem reconciled with a 'bottom-up' ecosystem for estimating effort at all levels of granularity, using the same FSM method. The COSMIC FP method can be used 'as is' for this purpose; other FSM methods can also be applied for use in Agile projects, albeit with adaptations of, or additions to, their standard rules.

More generally, software development practitioners would benefit from a) developing a better understanding of what are NFR, and b) giving greater priority to eliciting NFR and thinking about how they will be allocated (to software or to non-software) *early* in a project before starting design and coding, especially for NFR that apply across a whole system.

References

[1] Albrecht, A., 'Measuring Application Development Productivity', IBM Application Development Symposium, Monterey, CA, October 14-19, 1979

[2] The 'IFPUG Counting Practices Manual'. The International Function Point User Group. See www.ifpug.org. (Version 4.3 is published as ISO/IEC 20926.)

[3] The 'NESMA FPA Counting Practices Manual'. See https://nesma.org. (The 2008 version is published as ISO/IEC 24570.)

[4] The 'MkII FPA Counting Practices Manual'. (Version 1.3.1 of the MkII FPA method is published as ISO/IEC 20968.)

[5] The 'COSMIC Measurement Manual', See www.cosmic-sizing.org. (The 2011 version is published as ISO/IEC 19761.)

[6] See for example: What Are Story Points and Why Do We Use Them? (mountaingoatsoftware.com)

[7] The Entity – Attribute – Measure Taxonomy, Buglione L., Ebert C., Estimation, Encyclopaedia of. Software Engineering, Taylor & Francis Publisher, June. 2012, ISBN: 978-1-4200-5977-9)

[8] Boehm, B., *(1981). Software Engineering Economics. Prentice-Hall. ISBN 0-13-822122-7.*

[9] Santillo, L., 'Error Propagation in Software Measurement and Estimation', 16th International Workshop on Software Measurement, Potsdam, Germany, 2006

[10] Frederick Taylor, 'The Principles of Scientific Management', 1911. Re-published by W.W. Norton & Company, 1967, ISBN 0-393-00398-1.

[11] ISO/IEC 14143-1:2019 Information technology — Software measurement — Functional size measurement — Part 1: Definition of concepts

[12] ISO/IEC/IEEE 24765:2010 Systems and software engineering—Vocabulary.

[13] 'Glossary of terms for Non-Functional Requirements and Project Requirements used in software project performance measurement, benchmarking and estimating' version 1.0, September 2015. See www.cosmic-sizing.org or www.ifpug.org .

[14] 'Software engineering — Standard for software nonfunctional size measurements', ISO/IEC 32430, www.iso.org

[15] Abran, A., 'IEEE 2430 Non-Functional Sizing Measurements: A Numerical Placebo', IEEE Software , 2021 | Volume: 38, Issue: 3.

Towards a Fast Cost Estimation Supported by Large Language Models

Sandro Hartenstein[1,2], Sidney Leroy Johnson[1], Andreas Schmietendorf[1,2]

sandro.hartenstein@hwr-berlin.de, s_johnson20@stud.hwr-berlin.de, andreas.schmietendorf@hwr-berlin.de

[1] Berlin School of Economics and Law, Alt-Friedrichsfelde 60, 10315 Berlin, Germany

[2] Otto-von-Guericke-University Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany

Abstract

This elaboration describes the conceptual identification of the capabilities of large language models for cost estimation. For this purpose, the possibilities of the models for size estimation according to COSMIC Function Point are explained and examined prototypically. A comparison of a manual measurement with a trained and untrained model shows the current state of AI capabilities in a concrete example. Finally, the next steps within the study are briefly discussed.

Keywords

Artificial intelligence, size measurement, function points, cost estimation

1. Introduction

Embrace the power of Artificial Intelligence (AI) in software development cost estimation and revolutionize the way you plan your projects. Traditional cost estimation methods often rely on manual effort and subjective judgment, leading to inconsistencies, inaccuracies, and delays. By integrating AI into your estimation process, you unlock a world of possibilities.

AI is able to bring objectivity, efficiency, and predictive capabilities to the table. Through advanced algorithms and machine learning techniques, AI analyzes historical data, project parameters, and industry trends to generate accurate cost estimates. It eliminates human biases and enhances decision-making by considering a multitude of factors simultaneously.

With AI, you can benefit from faster estimation cycles, saving valuable time and resources. Its ability to process vast amounts of data in seconds means you can generate reliable cost estimates in a fraction of the time it would take using traditional methods. This efficiency allows you to allocate your resources more effectively, optimize project planning, and make informed decisions swiftly.

Moreover, AI-powered cost estimation provides a proactive approach. By leveraging machine-learning algorithms, AI continuously learns and adapts to new data, refining its estimation accuracy over time. It identifies patterns, recognizes project risks, and provides valuable insights that help you mitigate challenges before they arise. This proactive approach empowers you to make strategic adjustments, improve cost control, and increase the likelihood of project success.

The aim is to determine the possibilities of AI for automating and optimizing the COSMIC FP method. For this purpose, we have formulated the following research questions, which are to be answered in the proposal:

1. How can the cost estimation process supported by AI?
2. Can the software size be accurately determined with AI and COSMIC FP?

2. State of the Art

This section briefly explains the most important terms in the context of the subject area and presents current interesting studies and approaches.

2.1. Important terms

AI, short for Artificial Intelligence, refers to the simulation of human intelligence in machines that are programmed to think, learn, and perform tasks that typically require human intelligence. It is a branch of computer science that focuses on developing intelligent systems capable of replicating or emulating human cognitive abilities.

AI encompasses a wide range of techniques, algorithms, and approaches, including machine learning, deep learning, natural language processing, computer vision, expert systems, and more. These techniques enable AI systems to analyze and interpret complex data, recognize patterns, make decisions, solve problems, and even engage in natural language conversations.

Machine learning is a core component of AI and involves training models with large amounts of data to recognize patterns and make predictions or decisions. Deep learning, a subset of machine learning, utilizes artificial neural networks inspired by the structure of the human brain to process and learn from complex data.

AI systems can be categorized into two types: narrow AI and general AI. Narrow AI refers to systems designed to perform specific tasks within a defined domain. For example, image recognition, voice assistants, and recommendation systems are all examples of narrow AI applications. On the other hand, general AI aims to possess human-like intelligence across a broad range of tasks and exhibit characteristics such as reasoning, learning, and adaptability.

AI has widespread applications across various industries and domains. It is used in healthcare for diagnosing diseases and assisting in treatment plans, in finance for fraud detection and algorithmic trading, in transportation for autonomous vehicles, in customer service for chatbots, in manufacturing for process optimization, and in many other fields.

The field of AI continues to advance rapidly, with ongoing research and development exploring new techniques, algorithms, and applications. As AI systems become more sophisticated, they hold the potential to revolutionize industries, improve efficiency, and solve complex problems, making a significant impact on society as a whole.

Software Requirements are a description of the features and functionalities that a software system must have. They specify what the software should do and how it should perform.

There are several types of software requirements, including functional requirements, non-functional requirements, and domain requirements. Functional requirements describe the specific behaviors or functions of the software system. They specify what the system should do. Non-functional requirements describe the qualities or characteristics of the system. They specify how well the system should do what it does. Domain requirements are specific to the domain or industry in which the software will be used. They may include legal or regulatory requirements, industry standards, or other constraints that must be met by the software.

COSMIC FP (Function Point) is a software development methodology that measures the size of a system in terms of its functional requirements. It provides a standardized way to measure and compare the size of different systems, regardless of their technology or programming language. COSMIC FP is based on the concept of function points, which are used to quantify the functionality of a software system. The methodology involves analyzing the system's functional requirements and mapping them to specific function points, such as user inputs, outputs, and inquiries. By using this approach, organizations can better understand the size and complexity of their systems, identify areas for improvement, and make more informed decisions about software development projects.

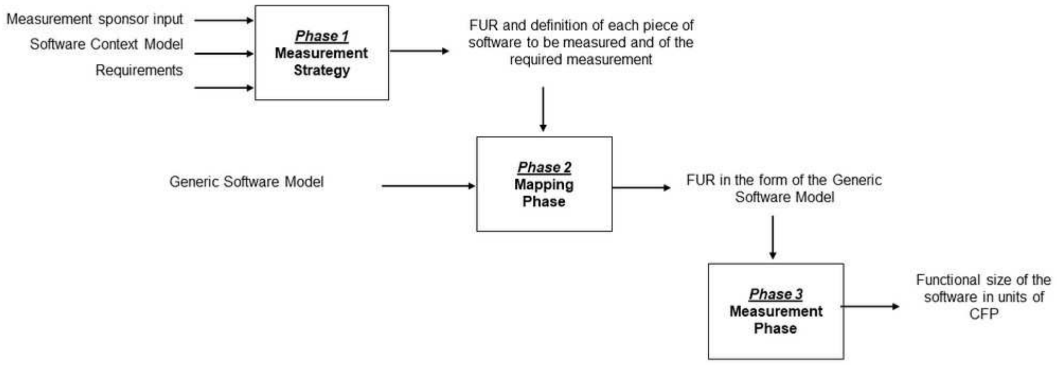
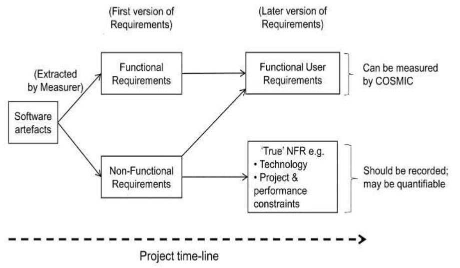


***Figure 1** Measurement process of COSMIC FP*

The measurement is carried out in three phases, which are shown in Figure 1 Measurement process of COSMIC FP . In the first phase, the functional user requirements, short FUR, are formed from the software context, requirements and measurement objectives. This is visualized in detail in . In the second phase, these are mapped to the Generic Software Model, which is shown in , so that the data movements become visible. In the third phase, the CFP of the individual data movements of the FUR are added together and result in the project size as CFP.



ure 2.4 - Many requirements initially appearing as NFR evolve into FUR as a project progresses.

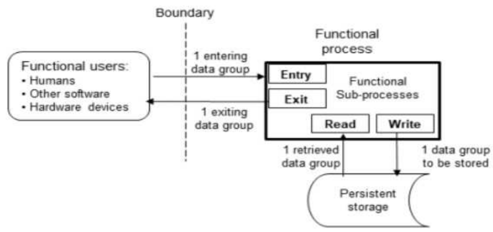


Figure – The four types of data movements

Figure 2 Functional User Requirements Figure 3 Generic Software Model

2.2. Related Studies

The requirement to determine the cost of software projects quickly and precisely has existed for a long time, but for an accurate prediction many parameters have to be taken into account. With AI Supported Software Engineering it is now also possible to automate software design. Three current studies serve as a technical preselection of the approach:

- This study from 2020 uses ensemble learning bagging with base learner Linear regression, SMOReg, MLP, random forest, REPTree, and M5Rule to estimate the cost of software development. The dataset is based on 499 projects. The results show that the Mean Magnitude Relative error of Bagging M5 rule with Genetic Algorithm as Feature Selection is 10%, which makes it better than other algorithms.
- This study from 2023 suggests a learning-based cost estimation model that leverages relational databases to improve accuracy. The proposed approach estimates project cost based on the effort required to complete software development, which is a key driver of the project cost. The proposed model is designed to address the challenges posed by the variability in open-source development, including variable team sizes, working hours and expertise. The proposed model is evaluated against 100 open source software repositories and shows its effectiveness in accurately estimating development costs.
- This study from 2021 involves research about software effort estimation using machine learning algorithms. The objective of this research is to use several algorithms of machine learning to estimate the effort of software project development. The best machine learning model is chosen to compare with the Constructive Cost Model (COCOMO) which is one of the well-established software project estimation models but has some weaknesses, but still has some weaknesses, including a lack of accuracy according to software developers)

3. Concept

Our approach is to estimate software size based on requirements catalogues and optimized AI models. The goal is the targeted support of Cosmic FP Analysis by AI for the identification of data movement types. Thus, phase 2 of the measurement process, shown in Figure 1 Measurement process of COSMIC FP , is to be automated. This information will then be used to carry out phase 3. Figure 4 Concept of a fast cost Estimation with AI and COSMIC FP illustrates the approach schematically.

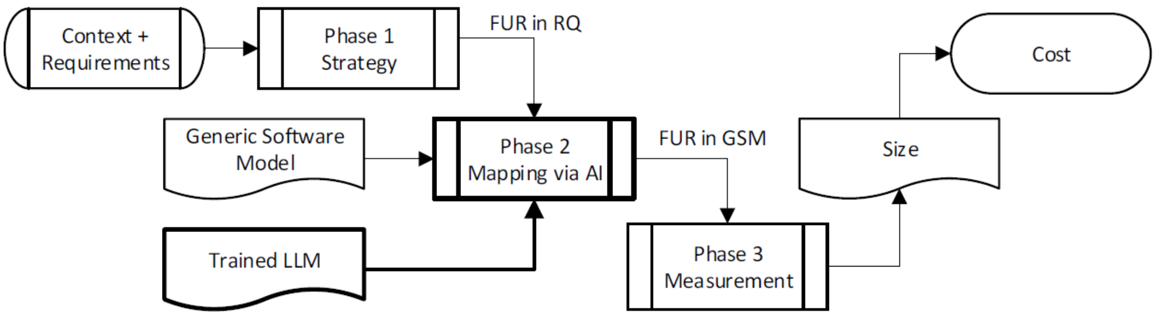


Figure 4 Concept of a fast cost Estimation with AI and COSMIC FP

With the size of the software product as CFP, established conversion factors for general types of software can be used to determine the anticipated costs. This conversion can be fine-tuned later using the software type and context. The conversion into person-months, or PM for short, is:

$$1 \text{ CFP} \approx 0{,}07 \text{ PM}$$

3.1. Capabilities of AI

In addition to looking at the current literature on the subject, we conducted our own investigations and tests with potential AI technologies. The focus was on the applicability and quality of the results for the concept.

A **large language model** (LLM) is a language model consisting of a neural network with many parameters (typically billions of weights or more), trained on large quantities of unlabeled text using self-supervised learning or semi-supervised learning. LLMs emerged around 2018 and perform well at a wide variety of tasks. They are deep learning neural networks, a subset of artificial intelligence and machine learning. Large language models are first pre-trained so that they learn basic language tasks and functions. Pretraining is the step that requires massive computational power and cutting-edge hardware. LLMs can recognize, summarize, translate, predict and generate text and other content based on knowledge gained from massive datasets3. They are among the most successful applications of transformer models.

In our consideration of LLMs, specifically ChatGPT, in the context of software development, we have potential in terms of generating software architectures, shown at the appendix 6.1. We found shortcomings in quality and accuracy, but the LLM was not trained for this focus. For this reason, LLMs trained specifically for the task could greatly improve cost estimation.

Semantic analysis is the process of using natural language processing (NLP), text analysis, and computational linguistics to identify and extract subjective information from source materials. It is used to determine the emotional tone behind a series of words, used to gain an understanding of the attitudes, opinions, and emotions expressed within an online mention. Some popular use cases of sentiment analysis include social media monitoring, customer support management, analyzing customer feedback, brand monitoring and reputation management

In our consideration of sematic analysis of software system requirements in the context of software engineering, we have identified potential. Furthermore, it was found that the quality of the results is very much dependent on the training data set.

3.2. Experiment

In order to determine the suitability of AI in measurement, a practical comparison was chosen as the method. This provides an initial picture of the performance and is manageable in terms of effort. Figure 5 Process for determining the suitability of LLM for sizing software projects. shows the workflow of the approach. The stages are as follows:

1. Context + Requirements
2. Legacy Measurement
3. AI Measurement
4. LLM Training
5. AI Measurement (Trained LLM)
6. Comparison

The first step is to define the context, i.e. the type and goal of the software, and to determine detailed requirements. This was done in a concrete experiment with the support of Chatgpt. The key results of this step are listed in Appendix 6.2.

In the second step, the measurement is carried out manually to obtain reference values. The results are listed in Appendix 6.3.

In the third step, the requirements are mapped to COSMIC FP using LLM and measured. The outputs are listed in Appendix 6.4.

The fourth step is the training of the LLM. Here, the LLM is prepared with a few partial results of the manual CFP mapping using promting. This is documented in Appendix 6.5.

In the fifth step, the requirements are mapped to COSMIC FP using the trained LLM and measured. These results are also shown in Appendix 6.6.

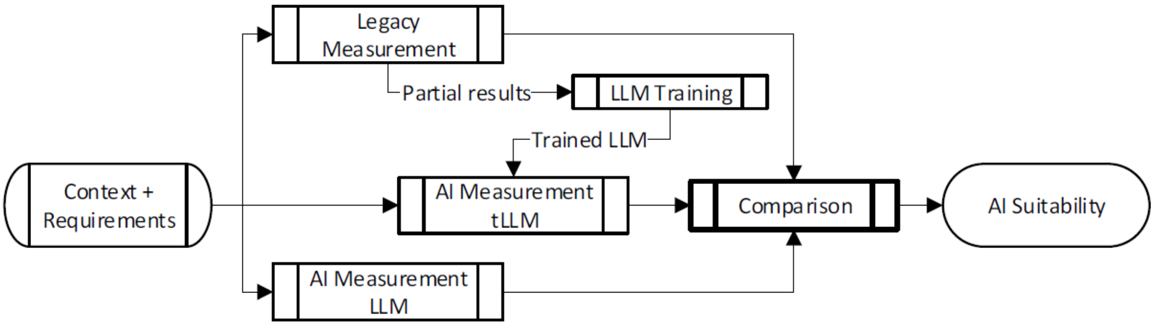


Figure 5 Process for determining the suitability of LLM for sizing software projects.

The criterion is the accuracy and the rough overall effort of the measurement.

3.3. Results

Table 1 Comparison of the measurements compares the results of the individual measurements in order to identify deviations of individual mapping points. An important finding is the close examination of the issue of the LLM. In the experiment, the first run classified and scored according to IFPUG FP, although COSMIC FP was requested. Furthermore, the LLM had to be made aware of missing results, e.g. writes, and the aggregation of the individual partial results was difficult. This could be addressed by using the intermediate results.

Table 1 Comparison of the measurements

	Legacy	LLM	tLLM
User Authentication and Authorization:	6	5	6
Contact Management	11	5	10
Lead and Opportunity Management:	10	5	12
Sales and Pipeline Management:	10	6	9
Customer Interaction Tracking:	6	5	7
Task and Activity Management:	6	5	7
Communication Integration:	7	7	7
Reporting and Analytics:	9	5	9
Integration with Other Systems:	8	6	6
Mobile Accessibility:	8	6	3
CFP	81	55	76
estimated Cost (PM)	5,67	3,85	5,32

The overall result shows a good suitability of LLMs. This is especially relevant against the background of the high effort required for manual mapping and measurement.

4. Conclusion

The results of the experiment with show clear potential for the use of AI in the sizing and cost estimation of software development projects, especially in the requirements phase. Which is at the beginning of the software development cycle and therefore no software artefacts are available yet. The results also show that the quality depends very much on the intensity of the training.

The findings in relation to the research questions are:

1. How can the cost estimation process be supported by AI?
    a. The cost estimation process can be greatly accelerated by using large scale models. Specifically, activities related to function user requirements, CFP mapping and CFP measurement can be automated.
    b. Communication with LLM is challenging and not always goal-oriented, as methods sometimes have to be demanded emphatically.
3. Can the software size be accurately determined with AI and COSMIC FP?
    c. The accuracy of the size measurement depends very much on the input parameters; if these are pre-trained with the help of promting, the result is much more accurate.
    d. Training the LLMs via promting is also non-trivial, as, for example, inaccurate examples of the LLM are exacerbated in their subsequent executions.

For further research on this topic, we are focusing on training (promting), especially on the development of cross-project datasets. Furthermore, a permanent validation of the input parameters with the AI outputs is helpful to address misunderstandings and false statements at an early stage.

5. References

[1] COMMON SOFTWARE MEASUREMENT INTERNATIONAL CONSORTIUM: *WHAT IS A COSMIC FUNCTION POINT?* URL https://cosmic-sizing.org/wp-content/uploads/2018/08/What-is-a-COSMIC-Function-Point-v1.0-1.pdf

[2] ABRAN, Alain ; FAGG, Peter ; LESTHERHUIS, Arlan: *COSMIC Measurement Manual for ISO 19761 : Part 2: Guidlines*

[3] ALHAZMI, Omar H. ; KHAN, Mohammed Zubair: *Software Effort Prediction Using Ensemble Learning Methods*. In: *Journal of Software Engineering and Applications* 13 (2020), Nr. 07, S. 143–160. URL https://www.scirp.org/pdf/jsea_2020071610583292.pdf

[4] NAGARAJAPPA, Ravi Kumar Begur ; SURESH, Yeresime: *Software Development Effort Estimation Using Relational Database and Optimized Learning Mechanism*. In: *Journal of Computer Science* 19 (2023), Nr. 4, S. 540–553. URL https://thescipub.com/pdf/jcssp.2023.540.553.pdf

[5] ZAKARIA, Noor Azura ; ISMAIL, Amelia Ritahani ; ALI, Afrujaan Yakath ; KHALID, Nur Hidayah Mohd ; ABIDIN, Nadzurah Zainal: *Software Project Estimation with Machine Learning* In: *International Journal of Advanced Computer Science and Applications* 12 (2021), Nr. 6

[6] DUMKE, Reiner (Hrsg.); SCHMIETENDORF, Andreas (Hrsg.); SEUFERT, Manfred (Hrsg.); WILLE Cornelius (Hrsg.): *Handbuch der Softwareumfangsmessung und Aufwandschätzung*. Berlin : Logos Verlag, 2014

[7] YANG, Jingfeng ; JIN, Hongye ; TANG, Ruixiang ; HAN, Xiaotian ; FENG, Qizhang ; JIANG, Haoming ; YIN, Bing ; HU, Xia: *Harnessing the Power of LLMs in Practice: A Survey on ChatGPT and Beyond*. URL https://arxiv.org/pdf/2304.13712.pdf

[8] HARTENSTEIN, Sandro ; SCHMIETENDORF, Andreas: KI basierte Veränderungen der Aufgaben innerhalb des Software Engineerings : Eine Bestandsaufnahme im Zusammenhang mit dem Hype um ChatGPT, Bd. 47. In: GI-Fachausschuß Management der Anwendungsentwicklung und -wartung (Hrsg.): *WI-MAW-Rundbrief* : *GI-Fachausschuß Management der Anwendungsentwicklung und -wartung*, 2023 (47), S. 3–13

[9] SCHMIETENDORF, Andreas ; HARTENSTEIN, Sandro ; JOHNSON, Sidney Leroy: KI-gestützte Modernisierung von Altanwendungen: (Sentiment-) Analysen im Diskurs des Anforderungsmanagements. In: GI-FG SRE (Hrsg.): *WSRE*, 2023 (25), S. 20–21

6. Appendix

6.1 Generic Software Model by ChatGPT

AN Provide a generic software model of a crm system

A generic software model of a CRM system can be represented using a layered architecture approach. Here is a simplified model:

1. Presentation Layer:
   - User Interface (UI): This layer provides the user interface for interacting with the CRM system. It includes components such as web pages, forms, and graphical elements.
   - User Input Handling: This component handles user inputs and interacts with other layers to process user requests.
2. Application Layer:
   - Business Logic: This layer contains the core business logic of the CRM system. It includes components for customer management, sales pipeline management, marketing automation, and customer support.
   - Workflow Management: This component manages the workflow and business processes within the CRM system, such as lead management, opportunity tracking, and customer service processes.
   - Integration Services: These services facilitate integration with other systems, such as email clients, social media platforms, and third-party applications.
3. Data Layer:
   - Database: This layer stores and manages the CRM system's data. It includes a database management system (DBMS) and tables for storing customer information, sales data, marketing campaigns, and support tickets.
   - Data Access: This component provides functionality to interact with the database. It includes services for querying, inserting, updating, and deleting data.
4. Infrastructure Layer:
   - Security: This component handles user authentication, authorization, and data security. It ensures that only authorized users can access the CRM system and that data is protected.
   - Performance and Scalability: This component optimizes system performance, monitors resource usage, and allows the CRM system to scale based on demand.
   - Integration Adapters: These adapters enable integration with external systems, such as CRM data synchronization with an ERP system or customer data import/export.
5. Analytics and Reporting Layer:
   - Reporting: This component generates reports and visualizations based on CRM data. It includes functionalities for creating custom reports, dashboards, and data visualizations.
   - Analytics: This component performs data analysis to gain insights into customer behavior, sales trends, marketing effectiveness, and customer support performance.

It's important to note that this is a simplified model, and the actual software model of a CRM system may vary depending on specific requirements, technologies used, and the complexity of the system.

Regenerate response

Generated under the use of ChatGPT (https://chat.openai.com/)

6.2 Requirements by ChatGPT

The entire request history can be found under: https://cloud.hwr-berlin.de/s/bmjWnHcBr3Fxsxb password: HYConMjzAj

Shorten results:

Question to ChatGPT:

Create a concrete software model of a CRM system for 1000 users and the main requirement of integrity, confidentiality and availability. It should include the typical requirements of a CRM system. Please provide all Functional User Requirements.

Answer from ChatGPT:

Sure! Here are some functional user requirements for a CRM system designed to cater to 1000 users with a focus on integrity, confidentiality, and availability:

1. User Authentication and Authorization:
   - The system should support secure user authentication and enforce strict access controls based on user roles and permissions.
   - Users should have unique credentials to access the system, and passwords should be securely stored using industry-standard encryption.
2. Contact Management:
   - Users should be able to add, view, edit, and delete contacts.
   - Contacts should include essential information such as name, email, phone number, address, and any additional custom fields.
   - The system should provide search and filtering capabilities to easily find and retrieve specific contacts.
3. Lead and Opportunity Management:
   - Users should be able to create and track leads and opportunities.
   - Each lead/opportunity should have relevant information such as source, status, value, expected closing date, and associated contacts.
   - Users should be able to assign leads/opportunities to specific users or teams for follow-up and collaboration.
   - The system should allow progress tracking, including updating the stage of each lead/opportunity.
4. Sales and Pipeline Management:
   - The system should provide functionality for managing the sales pipeline.
   - Users should be able to track the progress of deals, view sales targets, and analyze sales performance.

   - The system should generate reports and visualizations to provide insights into the sales process and identify potential bottlenecks.
5. Customer Interaction Tracking:
   - Users should be able to log and track all interactions with customers, including meetings, calls, emails, and notes.
   - The system should provide a centralized history of customer interactions for easy reference and follow-up.
6. Task and Activity Management:
   - Users should be able to create and manage tasks and activities related to contacts, leads, opportunities, and other CRM entities.
   - The system should support assigning tasks to users, setting due dates, and sending notifications/reminders.
7. Communication Integration:
   - The system should integrate with email clients to facilitate seamless email communication within the CRM.
   - Users should be able to send and receive emails directly from the CRM system and have them automatically linked to relevant contacts or leads.
8. Reporting and Analytics:
   - The system should offer comprehensive reporting capabilities, allowing users to generate predefined and custom reports.
   - Reports should cover various aspects, including sales performance, lead conversion rates, customer satisfaction, and team productivity.
   - The system should provide interactive dashboards and visualizations to analyze data effectively.
9. Integration with Other Systems:
   - The CRM system should support integration with other business applications such as marketing automation, customer support, or ERP systems.
   - Integration should enable seamless data sharing and synchronization between different systems.
10. Mobile Accessibility:
    - The CRM system should have a mobile-friendly interface or a dedicated mobile app to allow users to access and manage CRM data on the go.

Remember that these requirements are just a starting point, and you may need to further refine and tailor them based on the specific needs and preferences of your organization.

6.3 Legacy Measurement

Table 2 COSMIC FP Measurement for CR

	Entries E	Exits X	Reads R	Writes W	Sum CFP	
User Authentication and Authorization:						6
Authentication	1	1	1	0	3	
Authorization	1	1	1	0	3	
Contact Management						11
User	1	1	1	1	4	
Contacts	1	1	1	1	4	
Filter	1	1	1	0	3	
Lead and Opportunity Management:						10
Leads	1	1	1	1	4	
Details	0	0	1	1	2	
Assigns	0	0	1	1	2	
Tracking	1	0	0	1	2	
Sales and Pipeline Management:						10
Sales	1	1	1	1	4	
Tracking	1	1	0	1	3	
Reports	1	1	1	0	3	
Customer Interaction Tracking:						6
Tracking	1	1	0	1	3	
History	1	1	1	0	3	
Task and Activity Management:						6
Tasks	1	1	1	1	4	
Assigns	0	0	1	1	2	
Communication Integration:						7
Email	1	1	1	1	4	
Links	1	1	1	0	3	
Reporting and Analytics:						9
Reports	1	1	1	0	3	
Productivity	1	1	1	0	3	
Dashboards	1	1	1	0	3	
Integration with Other Systems:						8
Integration	1	1	1	1	4	
Sharing	1	1	1	1	4	
Mobile Accessibility:						8
Mobile App	1	1	3	3	8	
Sum CFP					**81**	

6.4. FUR Mapping by ChatGPT

The entire request history can be found under: https://cloud.hwr-berlin.de/s/bmjWnHcBr3Fxsxb password: HYConMjzAj

Shorten results:

Question to ChatGPT:

Please provide all Functional User Requirmentns and data movements (Entries, Exits, Reads, Writes) for a COSMIC FP analysis.

Answer form ChatGPT:

To perform a COSMIC (Counting Software Intensive Projects Using IFPUG Function Points) FP analysis, we need to define the functional user requirements and identify the data movements (entries, exits, reads, writes) associated with each requirement. Here are the functional user requirements and their corresponding data movements:

User Authentication and Authorization:
Data Movements:
Entries: User credentials (username, password)
Reads: User roles and permissions
Writes: N/A
Exits: Access granted or denied

Contact Management:
Data Movements:
Entries: New contact details (name, email, phone number, address, etc.)
Reads: Contact information for viewing or searching
Writes: Updated contact information
Exits: Confirmation of contact creation, retrieval, or update

Lead and Opportunity Management:
Data Movements:
Entries: New lead/opportunity details (source, status, value, etc.)
Reads: Lead/opportunity information for viewing or searching
Writes: Updated lead/opportunity information
Exits: Confirmation of lead/opportunity creation, retrieval, or update

Sales and Pipeline Management:
Data Movements:
Entries: Sales targets, progress updates
Reads: Sales performance data, pipeline status
Writes: N/A
Exits: Sales reports, pipeline analysis

Customer Interaction Tracking:
- Data Movements:
  - Entries: Logged interactions (meetings, calls, emails, notes)
  - Reads: Interaction history for specific customers
  - Writes: N/A
  - Exits: Interaction details, summary reports

Task and Activity Management:
- Data Movements:
  - Entries: New tasks or activities (description, due date, assigned user)
  - Reads: Task/activity details for viewing or searching
  - Writes: Updated task/activity information
  - Exits: Confirmation of task/activity creation, retrieval, or update

Communication Integration:
- Data Movements:
  - Entries: Email content, recipients
  - Reads: Received emails, linked contacts/leads
  - Writes: Sent emails, linked contacts/leads
  - Exits: Email sending/receiving status, linked email history

Reporting and Analytics:
- Data Movements:
  - Entries: Report parameters (filters, time periods, data sources)
  - Reads: Relevant data for generating reports
  - Writes: N/A
  - Exits: Generated reports, visualizations

Integration with Other Systems:
- Data Movements:
  - Entries: Data to be shared or synchronized with other systems
  - Reads: Data from other integrated systems
  - Writes: Data updates to other integrated systems
  - Exits: Synchronization confirmation, shared data status

Mobile Accessibility:
- Data Movements:
- Entries: User login credentials, mobile device information
- Reads: CRM data for viewing or manipulation
- Writes: Updated CRM data
- Exits: Confirmation of data access or updates via mobile devices

These are general data movements associated with each functional user requirement. It's important to note that the actual data movements may vary depending on the specific implementation and design of the CRM system.

6.5. CFP Measurement by ChatGPT

Entire request history: https://cloud.hwr-berlin.de/s/YraAoBnECefeayR password: pH65oAgyrW

Shorten results:

To estimate the Cosmic Function Points (FP) for the given CRM system functionalities, we will consider the data movements involved in each functionality. Cosmic FP is a measure of the functional size of a software system, considering the inputs, outputs, inquiries, files, and external interfaces. Each data movement category is assigned a weight, and we calculate the total unadjusted function points (UFP) by summing up the weights of the data movements.

Here is the estimation for each functionality:

1. User Authentication and Authorization:
   Data Movements:
   - Entries: 2 (user credentials)
   - Reads: 2 (user roles and permissions)
   - Writes: 0
   - Exits: 1 (access granted or denied)

   Total UFP for User Authentication and Authorization: 5

2. Contact Management:
   Data Movements:
   - Entries: 1 (new contact details)
   - Reads: 2 (contact information)
   - Writes: 1 (updated contact information)
   - Exits: 1 (confirmation of contact creation, retrieval, or update)

   Total UFP for Contact Management: 5

3. Lead and Opportunity Management:
   Data Movements:
   - Entries: 1 (new lead/opportunity details)
   - Reads: 2 (lead/opportunity information)
   - Writes: 1 (updated lead/opportunity information)
   - Exits: 1 (confirmation of lead/opportunity creation, retrieval, or update)

   Total UFP for Lead and Opportunity Management: 5

4. Sales and Pipeline Management:
   Data Movements:
   - Entries: 2 (sales targets, progress updates)
   - Reads: 2 (sales performance data, pipeline status)
   - Writes: 0
   - Exits: 2 (sales reports, pipeline analysis)

Total UFP for Sales and Pipeline Management: 6

5. Customer Interaction Tracking:
Data Movements:
- Entries: 1 (logged interactions)
- Reads: 2 (interaction history)
- Writes: 0
- Exits: 2 (interaction details, summary reports)

Total UFP for Customer Interaction Tracking: 5

6. Task and Activity Management:
Data Movements:
- Entries: 1 (new tasks or activities)
- Reads: 2 (task/activity details)
- Writes: 1 (updated task/activity information)
- Exits: 1 (confirmation of task/activity creation, retrieval, or update)

Total UFP for Task and Activity Management: 5

7. Communication Integration:
Data Movements:
- Entries: 2 (email content, recipients)
- Reads: 2 (received emails, linked contacts/leads)
- Writes: 1 (sent emails, linked contacts/leads)
- Exits: 2 (email sending/receiving status, linked email history)

Total UFP for Communication Integration: 7

8. Reporting and Analytics:
Data Movements:
- Entries: 1 (report parameters)
- Reads: 2 (relevant data for generating reports)
- Writes: 0
- Exits: 2 (generated reports, visualizations)

Total UFP for Reporting and Analytics: 5

9. Integration with Other Systems:
Data Movements:
- Entries: 1 (data to be shared or synchronized with other systems)
- Reads: 2 (data from other integrated systems)
- Writes: 1 (data updates to other integrated systems)
- Exits: 2 (synchronization confirmation, shared data status)

Total UFP for Integration with Other Systems: 6

10. Mobile Accessibility:

Data Movements:

- Entries: 2 (user login credentials, mobile device information)
- Reads: 2 (CRM data)
- Writes: 1 (updated CRM data)
- Exits: 1 (confirmation of data access or updates via mobile devices)

Total UFP for Mobile Accessibility: 6

Now, to calculate the total Cosmic FP, we sum up the UFP for each functionality:

Total Cosmic FP = 5 + 5 + 5 + 6 + 5 + 5 + 7 + 5 + 6 + 6 = **55**

Therefore, the estimated Cosmic Function Points (FP) for the given CRM system is 55.

6.6. FUR Mapping and CFP Measurement by trained ChatGPT

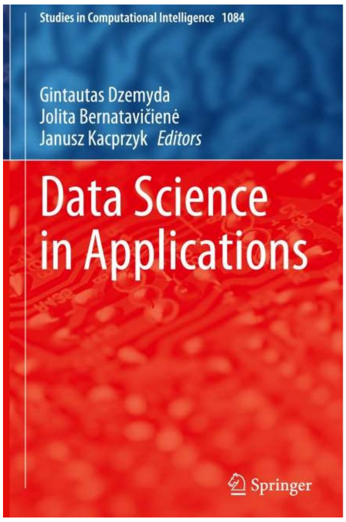
Entire request history: https://cloud.hwr-berlin.de/s/5N7jQzf2oD8MZYw password: rNYwtYWQoc

Grouped Results:

1. User Authentication and Authorization:
   Requirement 1: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Requirement 2: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Sum of Group 1: 6 CFP
2. Contact Management:
   Requirement 1: Entries: 1 Exits: 1 Reads: 1 Writes: 1 Sum: 4 CFP
   Requirement 2: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Requirement 3: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Sum of Group 2: 10 CFP
3. Lead and Opportunity Management:
   Requirement 1: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Requirement 2: Entries: 1 Exits: 1 Reads: 1 Writes: 1 Sum: 4 CFP
   Requirement 3: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Requirement 4: Entries: 0 Exits: 0 Reads: 1 Writes: 1 Sum: 2 CFP
   Sum of Group 3: 12 CFP
4. Sales and Pipeline Management:
   Requirement 1: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Requirement 2: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Requirement 3: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Sum of Group 4: 9 CFP
5. Customer Interaction Tracking:
   Requirement 1: Entries: 1 Exits: 1 Reads: 1 Writes: 1 Sum: 4 CFP
   Requirement 2: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Sum of Group 5: 7 CFP
6. Task and Activity Management:
   Requirement 1: Entries: 1 Exits: 1 Reads: 1 Writes: 1 Sum: 4 CFP
   Requirement 2: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Sum of Group 6: 7 CFP
7. Communication Integration:
   Requirement 1: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Requirement 2: Entries: 1 Exits: 1 Reads: 1 Writes: 1 Sum: 4 CFP
   Sum of Group 7: 7 CFP
8. Reporting and Analytics:
   Requirement 1: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Requirement 2: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Requirement 3: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Sum of Group 8: 9 CFP
9. Integration with Other Systems:
   Requirement 1: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Requirement 2: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
   Sum of Group 9: 6 CFP

10. Mobile Accessibility:
  Requirement 1: Entries: 1 Exits: 1 Reads: 1 Writes: 0 Sum: 3 CFP
  Sum of Group 10: 3 CFP

Sum of all CFP: 78 CFP

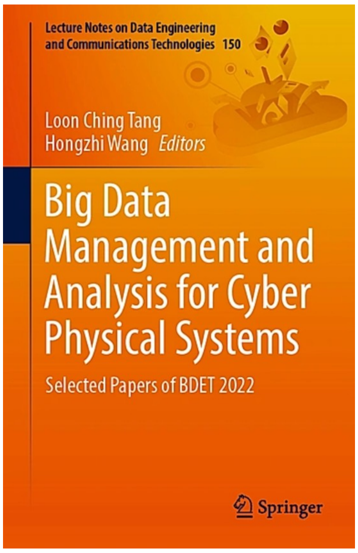


Gintautas Dzemyda et al.

Data Science in Applications

Springer Publ. 2023

The author provides an overview of a wide range of relevant applications and reveals how to solve them. Current applications in finance, technology, education, medicine and other important and relevant fields are data-driven with an enormous volumes of data. Specific methods need to be developed or adapted to solve a particular problem. It illustrates data science in applications. The set of examples in this book helps to solve essential problems for data management. This book can be used as material for lectures at universities.

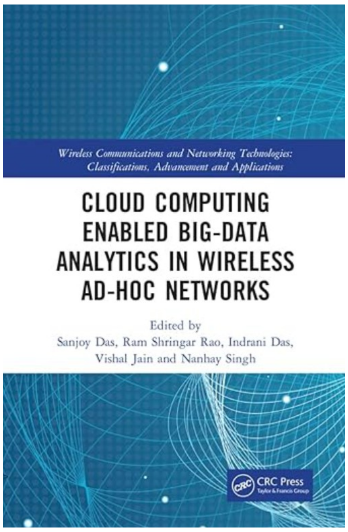


Loon Ching Tang, Hongzhi Wang:

Big Data Management and Analysis for Cyber Physical Systems

Springer Publ. 2022

This book consists of selected and peer-reviewed papers presented at 2022 4th International Conference on Big Data Engineering and Technology (BDET). The BDET conference series aims to provide the big data engineering and technology to inform the international colleagues in big data, algorithm and applications, emerging standards for big data, big data infrastructure, MapReduce and cloud computing, big data visualization, big data semantics, scientific discovery and intelligence, especially of the cyber-physical systems of interest.

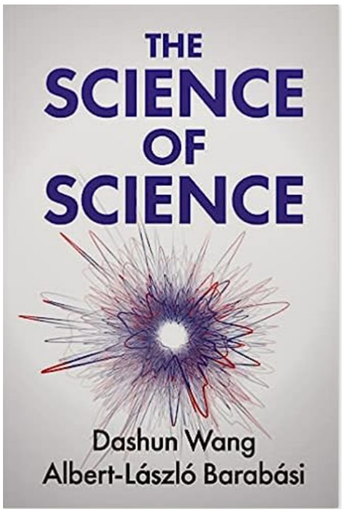


Sanjoy Das:

Cloud Computing Enabled Big-Data Analytics in Wireless Ad-hoc Networks (Wireless Communications and Networking Technologies)

CRC Press, 2022

The reference text covers big data concept in cloud based Wireless Ad hoc network and machine learning approaches in a single volume. It will be an ideal reference text for graduate students and academic researchers in the fields of electrical engineering, electronics and communication engineering, computer science and engineering.

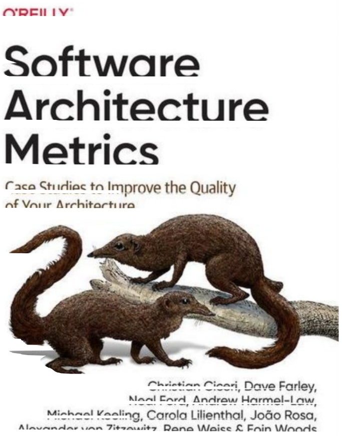


D. Wang, A.-L. Barabási:

The Science of Science Big Data, Metrics, and Impact

Cambridge University Press, 2021

„Big data analysis and quantitative tools help identify success and failure within the discipline. Areas in the 'science of science' that are ripe for further research are explored, and the implications this could have for future technological and innovative work are examined. With anecdotes and detailed, easy-to-follow explanations of the research, this book is accessible to all scientists, policy makers, and administrators with an interest in the wider scientific enterprise.“

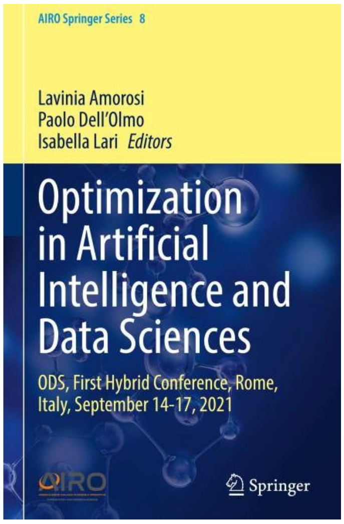


C. Cicero et al.:

Software Architecture Metrics

O'Reilly Publ, May 2022

„This isn't a book about theory. It's more about practice and implementation, about what has already been tried and worked. Detecting software architectural issues early is crucial for the success of your software: it helps mitigate the risk of poor performance and lowers the cost of repairing those issues. Written by practitioners for software architects and software developers eager to explore successful case studies, this guide will help you learn more about decision and measurement effectiveness.“

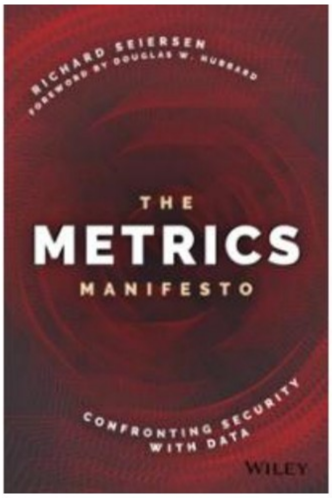


L. Amorosi, P. Dell'Olmo, I. Lari:

Optimization in Artificial Intelligence and Data Science

Springer Publ. Berlin, Heidelberg, 2022

„The book offers new and original contributions on different methodological optimization topics, from Support Vector Machines to Game Theory Network Models, from Mathematical Programming to Heuristic Algorithms, and Optimization Methods for a number of emerging problems from Truck and Drone delivery to Risk Assessment, from Power Networks Design to Portfolio Optimization. The articles in the book can give a significant edge to the general themes of sustainability and pollution reduction, distributive logistics, healthcare management in pandemic scenarios and clinical trials, distributed computing, scheduling, and many others.“

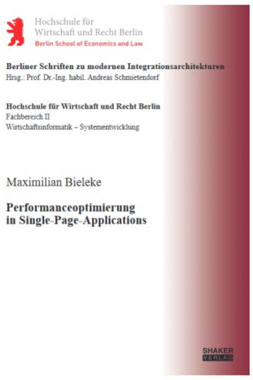


Seiersen, R.:

The Metrics Manifesto: Confronting Security with Data

John Wiley Publ., 2022, ISBN 978-1-119-51536-4

The Metrics Manifesto considers security with data delivers an examination of security metrics with R, the popular opensource programming language and software development environment for statistical computing. This insightful and upto-date guide offers readers a practical focus on applied measurement that can prove or disprove the efficacy of information security measures taken by a firm. The book's detailed chapters combine topics like security, predictive analytics, and R programming to present an authoritative and innovative approach to security metrics

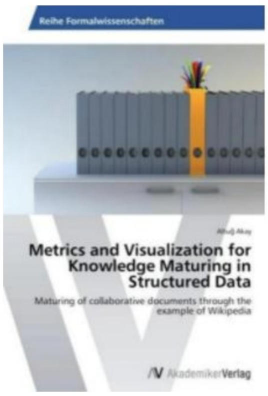


Maxemilian Bieleke:

Performanceoptimierung in Single-Page Application

Shaker-Verlag, Aachen, 2021, ISBN 978-3-8440-8315-6

Das vorliegende Buch beschreibt die Effizienz von Web-Applikationen hinsichtlich deren Performance in ausgewählten Anwendungs-bereichen.

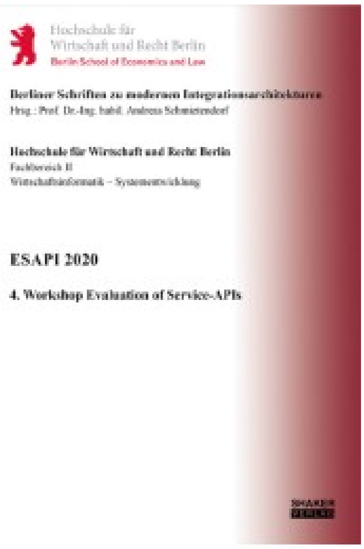


Akay, A.:

Metrics and Visualization for Know-ledge Maturing in Structured Data

Akademiker-Verlag, 2021

This book considers the maturing of information in collaborative environments such as Wikis, intranet documents or documents in cloud is investigated via four metrics. After the definition and calculation of the metrics, the results are visualized in graphical format. Therefore, the readers can see the evolution of the metrics within the time, but also the relations of metrics with each other.

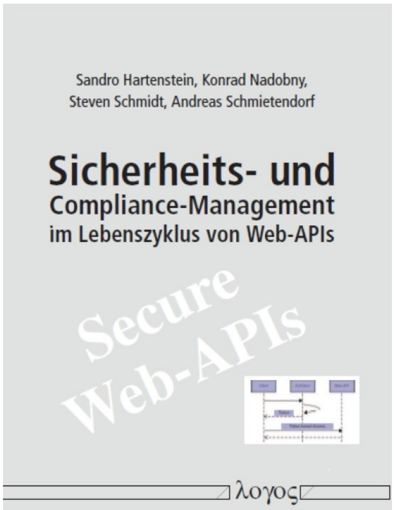


Andreas Schmietendorf

ESAPI 2020 – 4. Workshop Evaluation of Service-APIs

Shaker-Verlag, Aachen, November 2020, ISBN 978-3-8440-7515-1

Das vorliegende Buch fasst die insgesamt 11 Beiträge und Diskussionen des 4. Workshop zur Bewertung von service-basierten APIs zusammen und ist in der Buchreihe der Schriften zu modernen Integrations-architekturen erschienen.

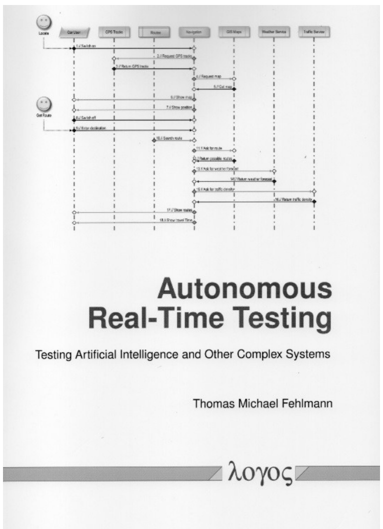


Hartenstein/Nadobny/Schmidt/ Schmietendorf:

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.



Thomas M. Fehlmann:

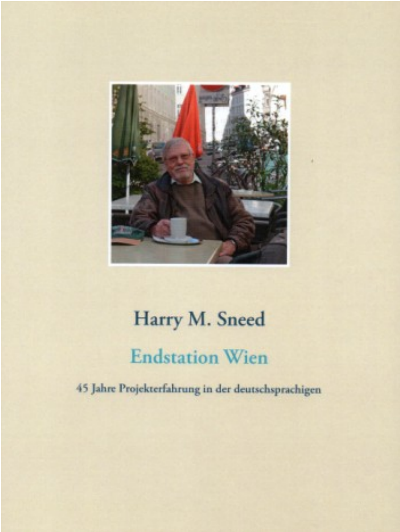
Autonomous Real-Time Testing

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.

Please remember:



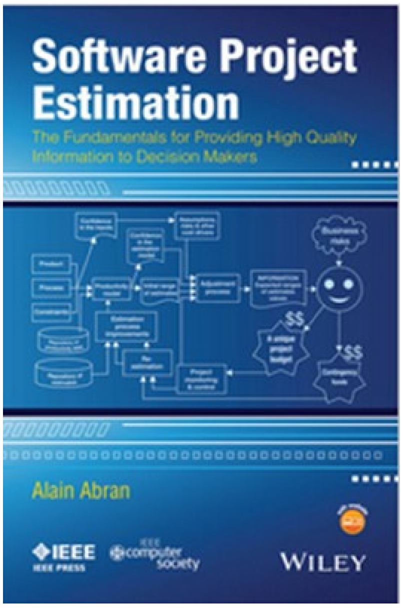
Harry Sneed:

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 Technologien und Systemen.

Dieses Buch fasst vor allem die umfangreichen Erfahrungen zu Wartungs-, Migrations- und Testprojekten zusammen, die auch für die Beherrschung aktueller und moderner Software-Anwendungen, von unschätzbarem Wert sind.



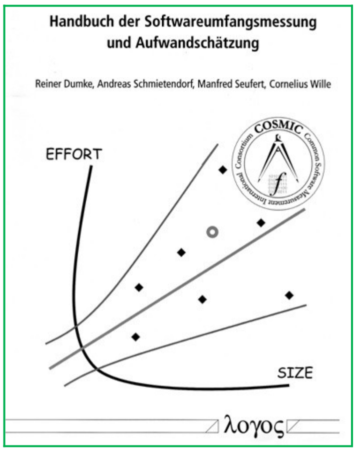
Abran, A.:

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.



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

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

September 2023

EuroAsiaSPI² 2023:	**European Systems & Software Process Improvement and Innovation Conference** August 30 – September 01, 2023, Grenoble, Switzerland **see: https://conference.eurospi.net/index.php/en/**
Smart Data Car Data 2023:	**Automobilwoche Konferenz** September 1, 2023, Munich, Germany **see: https://www.smartdata-cardata.de/programm.html**
Euromicro DSD/ SEAA 2023:	**Software Engineering & Advanced Application Conference** September 6 - 8, 2023, Durres, Albania **see: https://dsd-seaa2023.com/**
RE 2023:	**IEEE International Requirement Engineering Conference** September 4 - 8, 2023, Hannover, Germany **see: https://conf.researchr.org/home/RE-2023**
ASE 2023:	**Automated Software Engineering** September 11 - 15, 2023, Kirchberg, Luxembourg **see: https://conf.researchr.org/home/ase-2023**
IWSM/MENSURA 2023:	**The Join Conference of the 32nd International Workshop on Software Measurement and the 17th International Confernce on Software Process and Product Measurement** September 14 – 15, 2023, Rome, Italy **see: https://www.iwsm-mensura.org/**
QEST 2023:	**International Conference on Quantitative Evaluation of Systems** September 18 - 23, 2023, Antwerp, Belgium **see: https://www.qest.org/qest2023**
AIMS 2023:	**International Conference on AI and Mobile Services** September 23 - 26, 2023, Honolulu, USA **see: https://www.servicesociety.org/aims**
SCC 2023:	**International Conferences on Service Computing** September 23 – 26, 2023, Honolulu, USA **see: https://servicesociety.org/scc**
YSDS 2023:	**Young Scientists and early-stage research in Data Science** September 27, 2023, Berlin, Germany **see: https://fg-data-science.gi.de/**
BigData 2023:	**AI & Big Data Congress** September 27 - 28, 2023, Barcelona, Spain **see: http://aicongress.barcelona/es/**

IMMM 2023:	**International Conference on Advances in Information Mining and Management** June 26 - 30, 2023, Nice, France **see: https://www.iaria.org/conferences2023/IMMM23.html**

October 2023

data2day 2023:	**Konferenz für Big Data, Data Science und Machine Learning** October 11 – 12, 2023, Karlsruhe, Germany **see: https://www.data2day.de/**
API 2023:	**API Conference 2023** October 16 - 18, 2023, Berlin, Germany **see: https://apiconference.net/berlin-de/**
ESEIW 2023:	**Empirical Software Engineering International Week** October 23 - 27, 2023, New Orleans, USA **see: https://conf.researchr.org/home/eseiw-2023**
ESEM 2023:	**Conference on Empirical Software Engineering and Measurement** October 23 - 27, 2023, New Orleans, USA **see: https://conf.researchr.org/home/esem-2023**
ASQT 2023:	**Arbeitskonferenz Softwarequalität, Test und Innovation** *--- not this year ---* **see: http://www.asqt.org/**

November 2023

SEFM 2023:	**International Conference on Software Engineering and Formal Methods** November 6 - 10, 2023, Eindhoven, Netherlands **see: https://sefm-conference.github.io/2023/**
BigDataSE 2023:	**IEEE International Conference on Big Data Science and Engineering** *--- not this year ---* **see: http://www.ieee-hust-ncc.org/2022/BigDataSE/**
ICSEA 2023:	**International Conference on Software Engineering Advances** November 13 - 17, 2023, Valencia, Spain **see: https://www.iaria.org/conferences2023/ICSEA23.html**

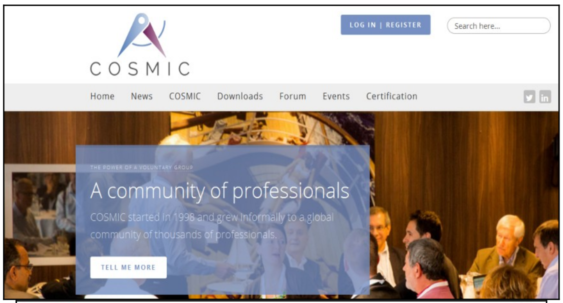
December 2023

IEEE ICDM 2023:	**IEEE International Conference on Data Mining** Dezember 1 - 4 , 2023, Shanghai, China **see: https://www.cloud-conf.net/icdm2023/index.html**
ESEC/FSE 2023:	**European Software Engineering Conference** and **Symposium on the Foundation of Software Engineering** Dezember 3 - 9, 2023, San Francisco, USA **see: https://2023.esec-fse.org/**
PROFES 2023:	**International Conference on Product Focused Software Process Improvement** Dezember 11 -13, 2023, Dornbirn, Australia **see: https://conf.researchr.com/home//profes-2023/**
BCD 2023:	**International Conference on Big Data, Cloud Computing, and Data Science Engineering** December 14 - 16, 2023, Hochimin City, Vietnam **see: https://acisinternational.org/conferences/bcd-2023/**
Big Data 2023:	**IEEE International Conference on Big Data** December 15-18, 2023. Sorrento, Italy **see: https://bigdataieee.org/BigData2023/**

see also:

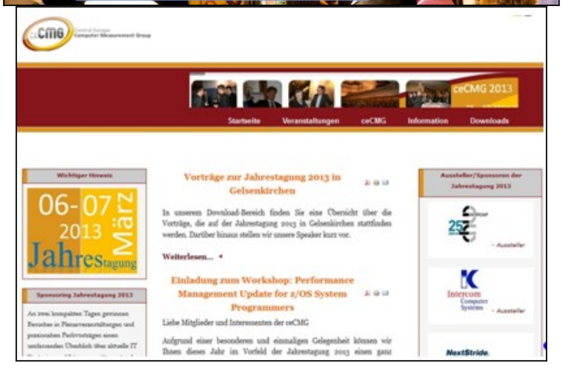
- http://**www.acisinternational.org/newconferences.html**
- **https://www.acm.org/conferences**
- **https://www.ieee.org/conferences_events/index.html**

COMMUNITIES



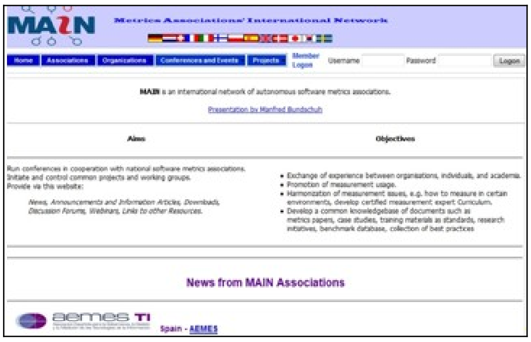
Common Software Measurement International Consortium (COSMIC)

http://cosmic-sizing.org



Central Europe Computer Measurement Group (ceCMG)

http://www.cecmg.de

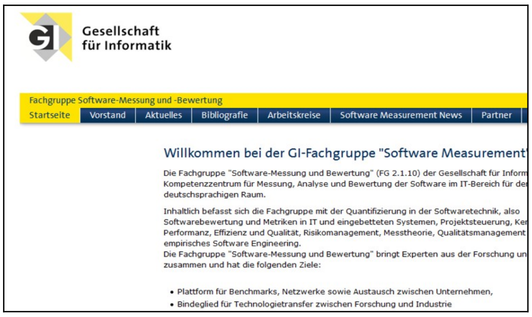


Metrics Association's International Network (MAIN)

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)

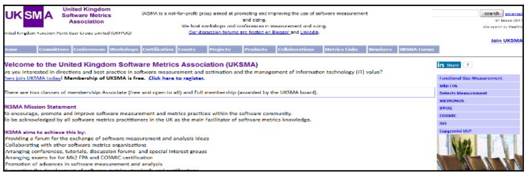
https://www.isbsg.org

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 Softwarequalität und Test (ASQT)

http://www.asqt.org

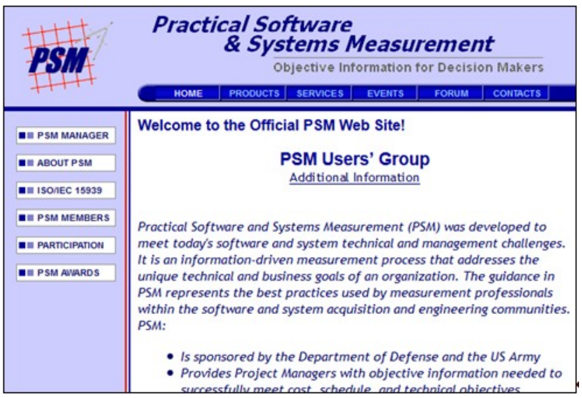
MEASUREMENT SERVICES

Software Measurement Laboratory (SML@b)

https://softmeasure.de

International Function Point Users Group (IFPUG)

http:www.ifpug.org

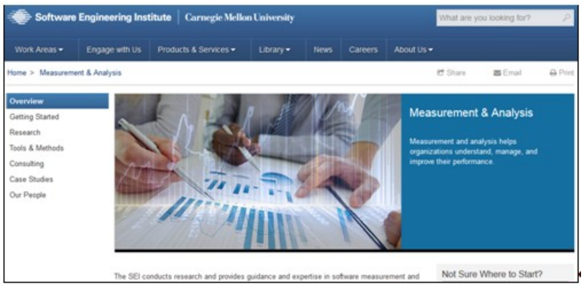


Practical Software & Systems Measurement

www.psmsc.com/:

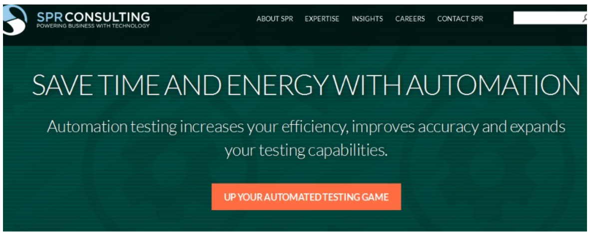
Computer Measurement Group (CMG)

http://www.cmg.org



Software Engineering Institute (SEI)

www.sei.cmu.edu/measurement/



Software Productivity Research (SPR)

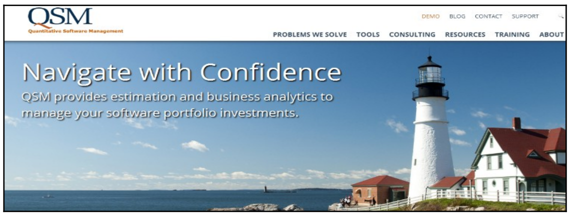
http://www.spr.com/

McCabe & Associates

http://www.mccabe.com

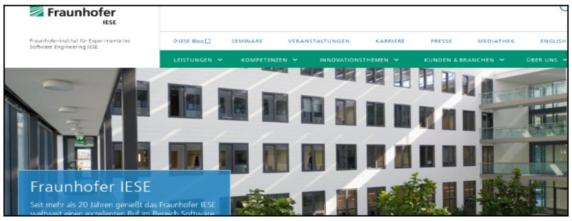
SQS Gesellschaft für Software-Qualitätssicherung

http://www.sqs.de



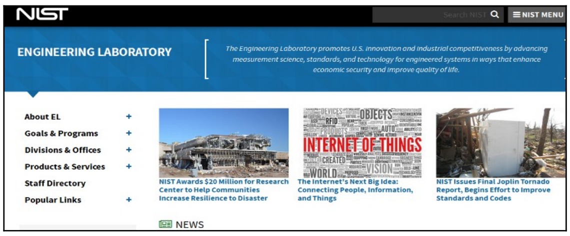
Quantitative Software Management (QSM)

http://www.qsm.com/



Fraunhofer Institute for Experimental Software Engineering (IESE)

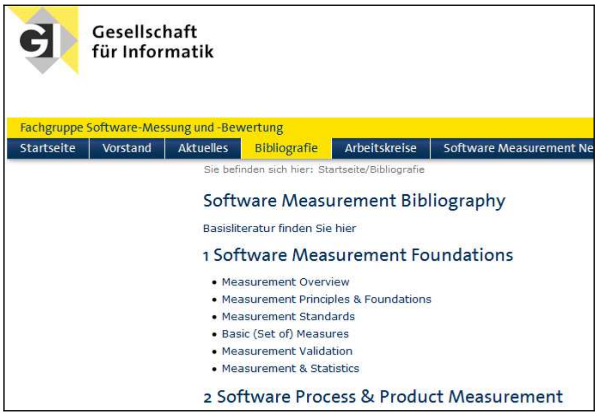
https://www.iese.fraunhofer.de/



National Institute of Standards and Technology (NIST)

https://www.nist.gov/el

SOFTWARE MEASUREMENT INFORMATION



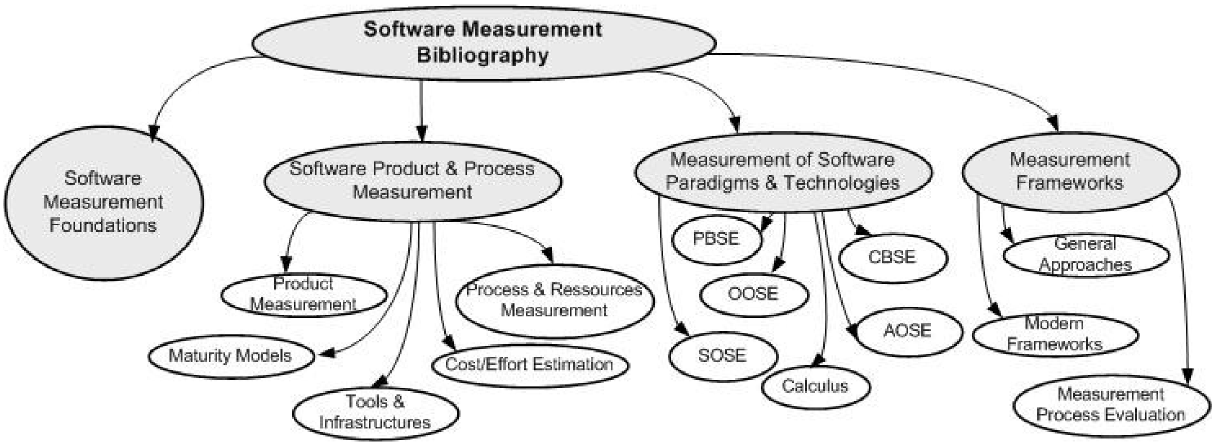
Software Measurement Bibliography

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

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

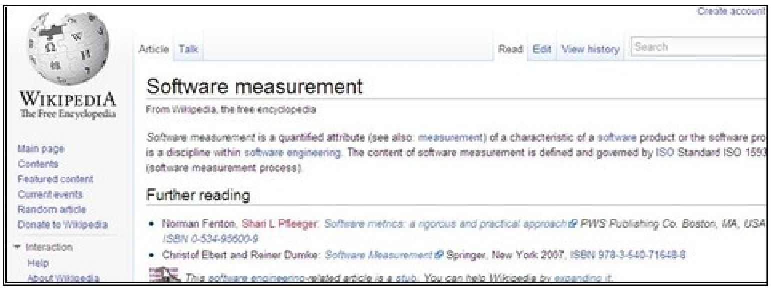
including any hundreds of books and papers

Bibliography Structure:



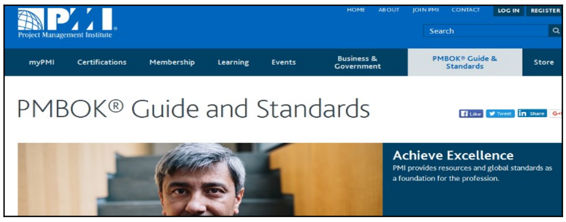
Software Measurement & Wikipedia

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 28 2023 NUMBER 2

CONTENTS

ISSN 1867-9196