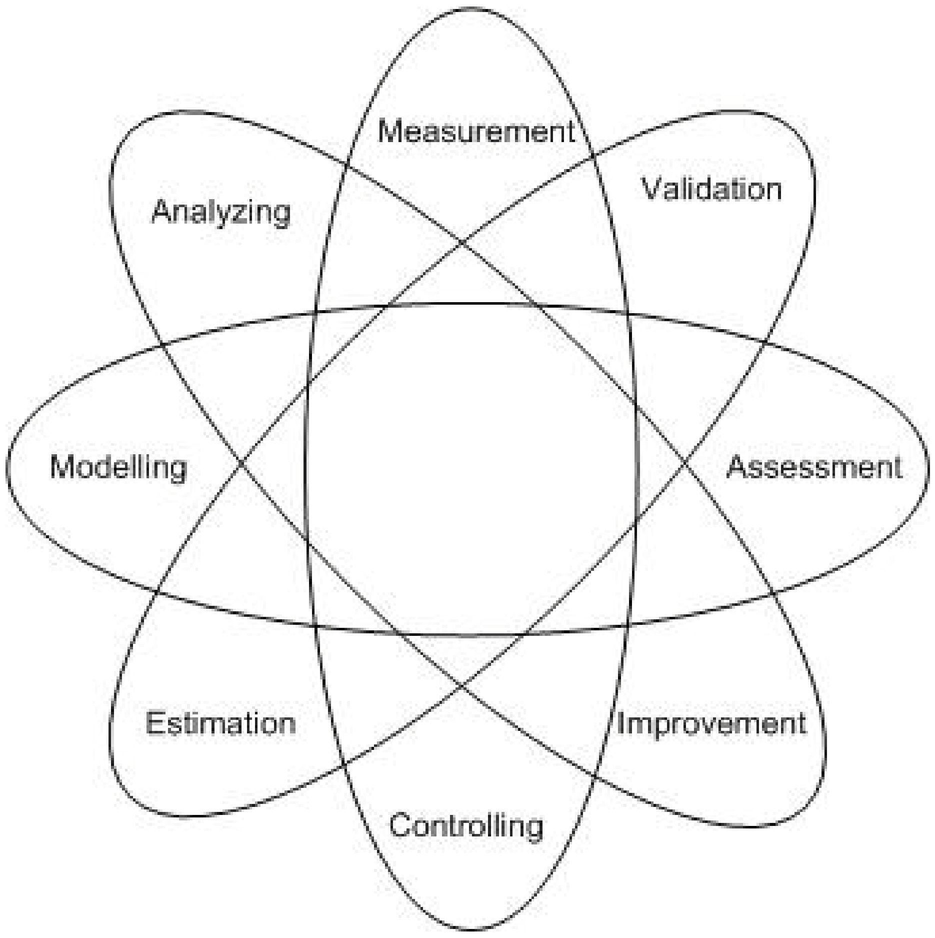
Volume 20, Number 1, February 2015

Software Measurement News

Journal of the Software Metrics Community



Editors:

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

Université du Québec
École de technologie supérieure

The *Software Measurement News* can be ordered directly from the Editorial Office (address can be found below).

Editors:

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

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

Reiner Dumke
Professor on Software Engineering
University of Magdeburg, FIN/IVS
Postfach 4120, D-39016 Magdeburg, Germany
***Tel.:** +49-391-67-52812*
dumke@ivs.cs.uni-magdeburg.de
http://www.smlab.de

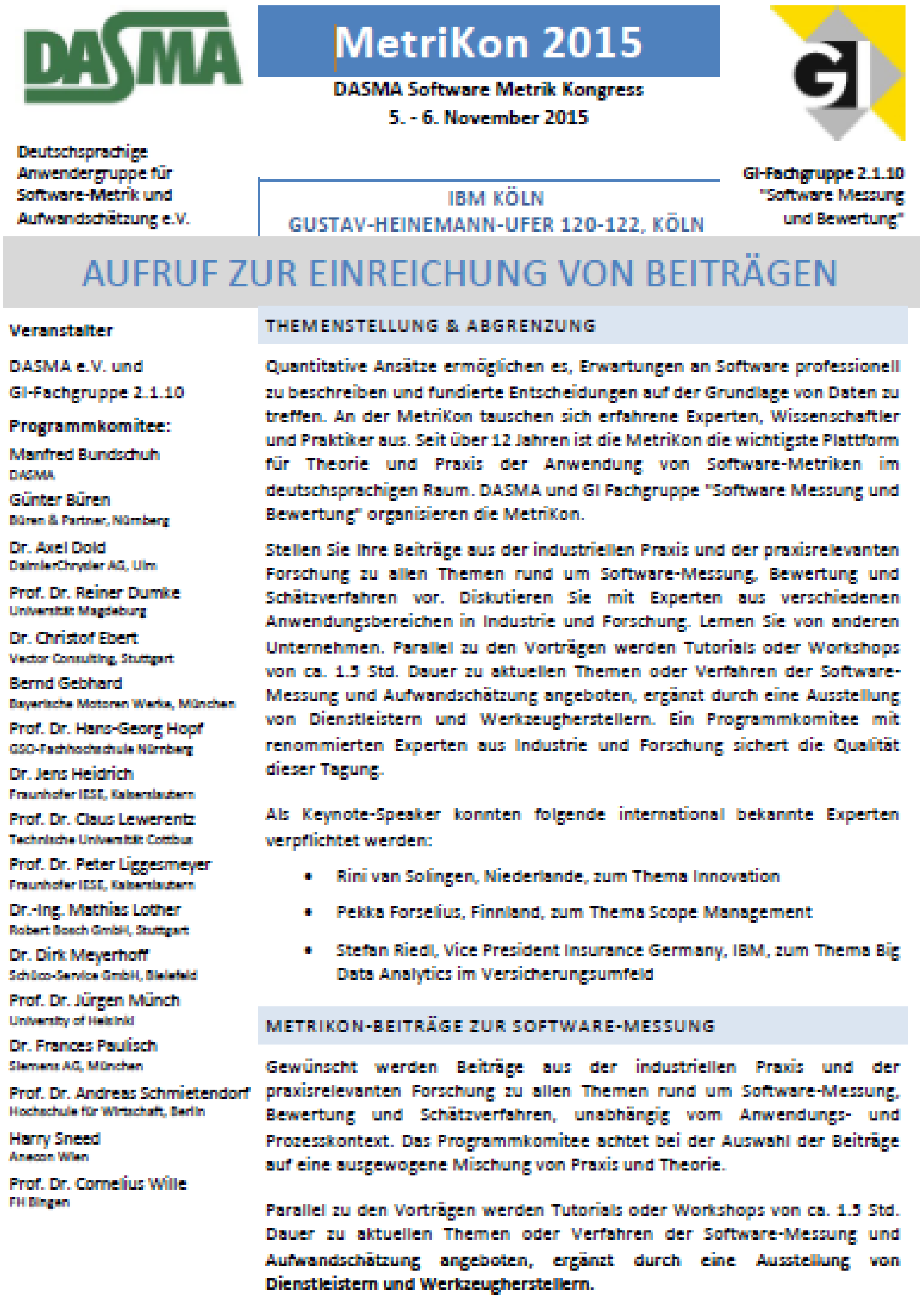
Christof Ebert
Dr.-Ing. in Computer Science
Vector Consulting GmbH
Ingersheimer Str. 20, D-70499 Stuttgart, Germany
***Tel.:** +49-711-80670-1525*
christof.ebert@vector.com

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

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

Technical Editor: Dagmar Dörge

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



MetriKon 2015

DASMA Software Metrik Kongress
5. - 6. November 2015

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

IBM KÖLN
GUSTAV-HEINEMANN-UFER 120-122, KÖLN

GI-Fachgruppe 2.1.10 "Software Messung und Bewertung"

AUFRUF ZUR EINREICHUNG VON BEITRÄGEN

Veranstalter

DASMA e.V. und
GI-Fachgruppe 2.1.10

Programmkomitee:

Manfred Bundschuh
DASMA

Günter Büren
Büren & Partner, Nürnberg

Dr. Axel Dold
DaimlerChrysler AG, Ulm

Prof. Dr. Reiner Dumke
Universität Magdeburg

Dr. Christof Ebert
Vector Consulting, Stuttgart

Bernd Gebhard
Bayerische Motoren Werke, München

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

Dr. Jens Heidrich
Fraunhofer IESE, Kaiserslautern

Prof. Dr. Claus Lewerentz
Technische Universität Cottbus

Prof. Dr. Peter Liggesmeyer
Fraunhofer IESE, Kaiserslautern

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

Dr. Dirk Meyerhoff
Schüco-Service GmbH, Bielefeld

Prof. Dr. Jürgen Münch
University of Helsinki

Dr. Frances Paulisch
Siemens AG, München

Prof. Dr. Andreas Schmietendorf
Hochschule für Wirtschaft, Berlin

Harry Sneed
Anecon Wien

Prof. Dr. Cornelius Wille
FH Bingen

THEMENSTELLUNG & ABGRENZUNG

Quantitative Ansätze ermöglichen es, Erwartungen an Software professionell zu beschreiben und fundierte Entscheidungen auf der Grundlage von Daten zu treffen. An der MetriKon tauschen sich erfahrene Experten, Wissenschaftler und Praktiker aus. Seit über 12 Jahren ist die MetriKon die wichtigste Plattform für Theorie und Praxis der Anwendung von Software-Metriken im deutschsprachigen Raum. DASMA und GI Fachgruppe "Software Messung und Bewertung" organisieren die MetriKon.

Stellen Sie Ihre Beiträge aus der industriellen Praxis und der praxisrelevanten Forschung zu allen Themen rund um Software-Messung, Bewertung und Schätzverfahren vor. Diskutieren Sie mit Experten aus verschiedenen Anwendungsbereichen in Industrie und Forschung. Lernen Sie von anderen Unternehmen. Parallel zu den Vorträgen werden Tutorials oder Workshops von ca. 1.5 Std. Dauer zu aktuellen Themen oder Verfahren der Software-Messung und Aufwandschätzung angeboten, ergänzt durch eine Ausstellung von Dienstleistern und Werkzeugherstellern. Ein Programmkomitee mit renommierten Experten aus Industrie und Forschung sichert die Qualität dieser Tagung.

Als Keynote-Speaker konnten folgende international bekannte Experten verpflichtet werden:

- Rini van Solingen, Niederlande, zum Thema Innovation
- Pekka Forselius, Finnland, zum Thema Scope Management
- Stefan Riedl, Vice President Insurance Germany, IBM, zum Thema Big Data Analytics im Versicherungsumfeld

METRIKON-BEITRÄGE ZUR SOFTWARE-MESSUNG

Gewünscht werden Beiträge aus der industriellen Praxis und der praxisrelevanten Forschung zu allen Themen rund um Software-Messung, Bewertung und Schätzverfahren, unabhängig vom Anwendungs- und Prozesskontext. Das Programmkomitee achtet bei der Auswahl der Beiträge auf eine ausgewogene Mischung von Praxis und Theorie.

Parallel zu den Vorträgen werden Tutorials oder Workshops von ca. 1.5 Std. Dauer zu aktuellen Themen oder Verfahren der Software-Messung und Aufwandschätzung angeboten, ergänzt durch eine Ausstellung von Dienstleistern und Werkzeugherstellern.

THEMENFELDER

Block 1: Projektmanagement:

- Messungen in Lean & Agilen Vorgehensweisen
- Schätzung von Initialaufwand und Restaufwand
- Scope Management
- Risikomanagement
- Fortschrittsverfolgung
- Projekt-Statusberichte
- Projekt-Kommunikation und Projekt-Marketing

Block 2: ICT Anwendungen:

- IT Controlling
- Messung und Benchmarks der Produktivität
- Benchmarking von Produkten, Prozessen und Projekten
- Portfolio-Management
- Automatisches Messen der funktionalen Größe
- Quantitative Verfahren für Qualitätssicherung und Qualitätsmanagement, z.B. Six Sigma
- Praktischer Einsatz von Mess- und Aufwandschätzverfahren in realen Anwendungen

Block 3: Messung und Bewertung:

- Erprobung, Einführung und praktische Anwendung von Software-Metriken
- Big Data – Metriken und Messverfahren
- Praktische Anwendung, z.B. für Funktionale Sicherheit, Codequalität und Test
- Governance und CMMI/SPICE/ITIL/COBIT-konforme Messprogramme
- Quantitatives Prozessmanagement und Optimierung für Produkte und Prozesse
- Messverfahren für Apps-Portfolio und mobile Plattformen

EINREICHUNGEN UND WEITERE INFORMATIONEN

Einreichung der MetriKon-Beiträge:

www.easychair.org

Ausstellungs-reservierungen:

info@dasma.org

Eine aussagekräftige Kurzfassung Ihres Beitrages/Vorschlages reichen Sie bitte formlos im pdf-Format über das Tagungsportal EasyChair ein (**http://www.easychair.org/conferences/?conf=metrikon2015**).

Alle für Autoren wichtigen Informationen und Vorlagen finden Sie auf der MetriKon-Website unter der unten angegebenen Adresse in der Rubrik „Beiträge".

Weitere Informationen unter **http://www.metrikon.de** (zur Tagung) bzw. **http://dasma.org** (zur DASMA e.V.) oder unter **http://fg-metriken.gi.de** (zur GI-Fachgruppe 2.1.10)

WICHTIGE TERMINE

22. Juni 2015:	Abgabeschluss einer aussagekräftigen Kurzfassung
5. Juli 2015:	Abgabeschluss für Workshops und Tutorials
bis 3. August 2015:	Benachrichtigung über die Annahme
28. August 2015:	Meldung von möglichen Ausstellern bei der Tagungsleitung
14. September 2015:	Abgabe des druckfertigen Beitrages

Call for Papers

IWSM Mensura 2015

ABOUT THE CONFERENCE

The IWSM Mensura conference is the conference where new ideas from the world of academic research meet practical improvements from industry on topics of measuring software. Each year practitioners and researchers from all over the world gather together to learn about new developments, test new ideas and exchange possible new solutions and applications.

On **October 5-7, 2015** the IWSM Mensura conference will be held in **Cracow, Poland**. The conference will take place in the Sheraton hotel in Cracow. More information on the conference can be found on the website: www.iwsm-mensura.org.

THEME & SCOPE

Software measurement techniques, methods, processes and tools are keys for successfully managing and controlling software development projects. Measurement is essential for any engineering activity and for increasing scientific and technical knowledge regarding both the practice of software development and empirical research in software technology.

The conference focuses on all aspects of software measurement and facilitates the exchange of software measurement experiences between theory and practice.

TOPICS OF INTEREST

We encourage submissions in any field of software measurement, including, but not limited to:

- Software measurement foundations
- Practical measurement applications
- Quantitative and qualitative methods for software measurement
- Measurement processes and resources
- Empirical case studies
- Measurement acceptance
- Enterprise embedded solutions of measurement
- Metrics validation
- Measurement for system and software engineering
- Measurement for integration, and testing
- Measurement for specific areas (e.g. ECU's or webservices)
- Measurement for specific development paradigms (e.g. agile or model-driven)
- Functional size measurement
- Software Measurement Standards
- Software estimation
- Software benchmarking
- Measurement services
- Measurement tools
- Measurement experience and guidance
- Theory of measurement
- Measurement paradigms

PAPERS

Papers must be submitted for review by the Program Committee through the EasyChair conference management system at: https://easychair.org/conferences/?conf=iwsmmensura2015

- Full papers (12 to 16 pages) or
- Short papers (6 to 10 pages)

Papers should not have already been published elsewhere, nor should they have been submitted to a journal or to another conference. At least one among the authors of each paper accepted must register for the conference and commit to paper presentation. All papers submitted must follow the Springer LNCS format.

Accepted and presented papers (full papers and short papers) will be included in the conference proceedings. Publication in Springer LNBIP (Lecture Notes in Business Information Processing) is applied (final decision pending).

WORKSHOP PROPOSALS

The main idea of the workshops is to bring both practitioners and researchers together to exchange ideas on particular topics of importance. Workshop proposals should be described on two A4 pages maximum and submitted directly to the Organization Chair via kobyl@sgh.waw.pl.

CONFERENCE LANGUAGE

The language for the conference, workshops and special sessions is English.

IMPORTANT DATES

	Full papers	Short papers	Workshop / Presentation
Submission	**May 3rd, 2015**	May 3rd, 2015	May 24th, 2015
Notification of acceptance	**June 12th, 2015**	June 12th, 2015	June 16th, 2015
Final version	**July 1st, 2015**	July 1st, 2015	August 31st, 2015

CONTACT INFORMATION PROGRAM CHAIRS

If you have any questions with regards to this call for papers you may contact the Program Chairs on kobyl@sgh.waw.pl.

General Chair for IWSM Mensura 2015

Andrzej Kobyliński (Warsaw School of Economics, Poland)

Program Co-chairs

Beata Czarnacka-Chrobot (Warsaw School of Economics, Poland, bczarn@sgh.waw.pl)
Jarosław Świerczek (Polish Software Measurement Association, Poland, jaroslaw.swierczek@psmo.pl)

Organization Committee

Grzegorz Poręcki (Polish Software Measurement Association, Poland, grzegorz.porecki@gmail.com)
Bogumiła Różyńska (Polish Software Measurement Association, Poland, bogumila.rozynska@gmail.com)

Conference fee will be 450 EURO. More information about registration process and early birds fee will be available soon on www.iwsm-mensura.org.

PROGRAM COMMITTEE

Silvia Abrahao	Universitat Politecnica de Valencia	Spain
Alain Abran	Ecole de Technologie Supérieure | UQAM	Canada
Mauricio Aguiar	TI Metricas	Brazil
Rafa Al-Qutaish	ÉTS, University of Quebec	Canada
Tiago Alves	Microsoft	Portugal
Sousuke Amasaki	Okayama Prefectural University	Japan
Luigi Buglione	École de Technologie Supérieure | Engineering.IT	Italy
Manfred Bundschuh	DASMA	Germany
Tom Cagley	David Consulting Group	United States
Laila Cheikhi	ENSIAS	Morocco
Marcus Ciolkowski	QAware GmbH	Germany
Beata Czarnacka-Chrobot	Warsaw School of Economics	Poland
Ton Dekkers	NESMA	the Netherlands
Onur Demirörs	Middle East Technical University	Turkey
Jean-Marc Desharnais	Ecole de Technologie Supérieure | UQAM	Canada
Reiner Dumke	University of Magdeburg	Germany
Christof Ebert	Vector Consulting	Germany
Thomas Fehlmann	Euro Project Office	Switzerland
Dan Galorath	Galorath	United States
Çigdem Gencel	Free University of Bolzano	Italy
Marcela Genero	University of Castilla-La Mancha	Spain
Naji Habra	PReCISE Research Center University of Namur	Belgium
Emilio Insfran	Universitat Politècnica de València (DSIC-UPV)	Spain
Jens Bæk Jørgensen	Mjølner Informatics A/S	Denmark
Alpay Karagöz	Innova IT Solutions	Turkey
Andrzej Kobyliński	Warsaw School of Economics	Poland
Rob Kusters	Eindhoven University Of Technology	the Netherlands
Luigi Lavazza	Università degli Studi dell'Insubria	Italy
Jean-Louis Letouzey	Inspearit	France
Mathias Lother	Robert Bosch GmbH	Germany
Beatriz Marín	Universidad Diego Portales	Chile
Roberto Meli	DPO	Italy
Arlene Minkiewicz	PRICE Systems	United Kingdom
Eduardo Miranda	Carnegie Mellon University	United States
Yoshiki Mitani	MITANI Advanced Research Institute	Japan
Akito Monden	NAIST	Japan
Maurizio Morisio	Politecnico di Torino	Italy
Makoto Nonaka	Toyo University	Japan
Rory O'Connor	Lero - the Irish software engineering research centre	Ireland
Olga Ormandjieva	Concordia University	Canada
Kai Petersen	Blekinge Institute of Technology	Sweden
Keith Phalp	Bournemouth University	United Kingdom
Geert Poels	Ghent University	Belgium
Grzegorz Poręcki	Polish Software Measurement Association	Poland
Rudolf Ramler	Software Competence Center Hagenberg GmbH	Austria
Gabriela Robiolo	Universidad Austral	Argentina
Andreas Schmietendorf	Berlin School of Economics and Law	Germany
Asma Sellami	University of Sfax	Tunisia
Martin Shepperd	Brunel University	United Kingdom
Miroslaw Staron	University of Gothenburg	Sweden
Charles Symons	COSMIC	United Kingdom
Jarosław Swierczek	Polish Software Measurement Association | 300DC	Poland
Ayça Tarhan	Hacettepe University	Turkey
Sylvie Trudel	Université du Québec à Montréal	Canada
Harold van Heeringen	Sogeti Nederland B.V.	the Netherlands
Monica Villavicencio	ESPOL	Ecuador
Stefan Wagner	University of Stuttgart	Germany
Dietmar Winkler	Vienna University of Technology	Austria
Chris Woodward	Chris Woodward Associates Ltd.	United Kingdom

Herausforderungen im Kontext von Big Data Lösungen

(Qualitative und quantitative Bewertung)

26.03.2015 (09:30 bis 17:00 Uhr) Hannover, Maritim-Hotel

Eine Kombination von Seminar, Workshop und Diskussionsrunde bietet Ihnen die Möglichkeit, sich schnell, praxisorientiert und interaktiv in die aktuellen Herausforderungen von Big Data-Lösungen einzuarbeiten. Während die ersten drei Beiträge Fachwissen in seminaristischer Weise vermitteln, bieten die Impulsvorträge der 4. Session die Möglichkeit der Anwendung bzw. Übertragung der erworbenen Kenntnisse auf konkrete Industrieprobleme.

Nach dem Besuch der Veranstaltung werden die Teilnehmer in der Lage sein, Einsatzszenarien für Big Data sowohl objektiv zu bewerten als auch kleinere Anforderungen einer konzeptionellen Lösung zuzuführen.

Eröffnung der Veranstaltung (09:30 Uhr):

Prof. Dr. Andreas Schmietendorf (HWR Berlin/Uni Magdeburg)
Big Data – Spannungsfeld zwischen Technik und Einsatzszenarien

- Frameworks (Hortonworks, Cloudera und Co)
- Implementierungsstrategien

Session 1 (10:00 bis 10:45 Uhr):

Dr. Robert Neumann, Ultra Tendency UG
Hadoop: Übersicht zum Framework

- Erkundung von Hadoop über die Konsole und den Browser
- Ausgewählte Praxisbeispiele

Session 2 (11:15 bis 12:00 Uhr):

Mitch Köhler, Cabalon
Column Family Database HBase

- Eigenschaften und Einsatzgebiete
- HBase & Hadoop MapReduce

Session 3 (13:30 bis 14:15 Uhr):

Jan Hentschel, Ultra Tendency UG
Document Database MongoDB

- Eigenschaften und Einsatzgebiete
- MongoDB & Hadoop MapReduce

Session 4 (14:15 bis 15:00 Uhr):

Frederik Kramer, *initOS GmbH & Co. KG*
In-Memory Computing mit SAP Hana

- Architektur – Konsequenzen für Anwender
- Implementierungsalternativen

Session 5 (15:30 bis 16:30 Uhr):

Joachim Kolbe, SYRACOM AG

Big Data in der Finanzindustrie (Praxisbericht)

Wolfgang Schwab, SAS Institute GmbH

Möglichkeiten Big Data Analytics (Praxisbericht)

Michael Weiß, HUK Coburg

Big Data in der Versicherungsindustrie (Praxisbericht)

Session 6 (16:30 bis 17:00 Uhr):

Markus Bauer, UFD AG, Andreas Schmietendorf, HWR Berlin

Moderierte Abschlussdiskussion

Die korrespondierenden Vorträge der Referenten werden den Teilnehmern in Form eines Handouts zur Verfügung gestellt. Ergebnisse entsprechender Diskussionsrunden werden zeitnah im Internet publiziert. Änderungen am Programm sind unter Vorbehalt möglich. Für Verpflegung vor Ort wird gesorgt. Jeder zahlende Teilnehmer erhält ein offizielles Zertifikat der ceCMG.

Für die Teilnahme an der Veranstaltung ist eine kostenpflichtige Anmeldung zur Enterprise Computing Conference (ECC 2015) erforderlich. Für Mitglieder der ceCMG-, DASMA-, GI- und ASQF gilt eine reduzierte Teilnahmegebühr. Über die Teilnahmegebühr erhalten Sie eine Rechnung der ceCMG e.V. (Central Europe Computer Measurement Group).

Veranstaltungsort: *Maritim Hotel Hannover (am Flughafen)*

Weiteren Informationen und Anmeldung unter: http://www.cecmg.de

Kontakt: Susanne Mund – sekretariat@cecmg.de

Bewertungsaspekte service- und cloudbasierter Architekturen (BSOA/BCloud2014) - detaillierter Workshopbericht

Andreas Schmietendorf[+], *Frank Simon*[#]

[+]Hochschule für Wirtschaft und Recht Berlin
Email: andreas.schmietendorf@hwr-berlin.de
[#]BLUECARAT AG
Email: frank.simon@bluecarat.de

1. Hintergründe zur Initiative BSOA/BCloud

Die ursprünglich im Zusammenhang mit serviceorientierten Architekturen gegründete Initiative beschäftigt sich mit der Bewertung vielfältig auftretender Integrationsanforderungen einer zunehmend digitalisierten und damit vernetzten Welt. Treiber dieser Entwicklung sind Themen wie Industrie 4.0 (Internet of Things), mobil eingesetzte Softwaresysteme, das Cloud-Paradigma oder auch analytisch eingesetzte Datenbanksysteme im Umfeld von Big Data. Unter Verwendung von Modellen, Methoden und konkreten Techniken gilt es anforderungsgerechte APIs herauszuarbeiten, welche als internetbasierte Serviceangebote bereitgestellt werden. Waren es in der Vergangenheit ausschließlich Softwareentwickler, die den API-Begriff in den Mittelpunkt einer kompositorischen Softwareentwicklung gestellt haben, werden APIs im wachsenden Maße mit der strategischen Unternehmensausrichtung in Verbindung gebracht, wie auch das folgende Zitat von [Spencer 2015] unterstreicht.

> „Application Programming Interfaces (API's) have gone from a something that only developers and architects once discussed to emerge as a capability that is central to many successful companies business strategies and a key focus of many of their senior leadership teams.”

Hintergrund dieser Tendenz ist die geforderte Fähigkeit eines Unternehmens, an unternehmensübergreifend und ggf. auch global ablaufenden Geschäftsprozessen agil teilhaben zu können. Dabei gilt es die Wirtschaftlichkeit und Qualität von Serviceangeboten über den gesamten Lebenszyklus sicherzustellen. Aus technologischer Sicht handelt es sich bei derartigen Services zumeist um RESTful Web Services, die mit vielfältigen Repräsentationsformen der im Internet verteilten Ressourcen umgehen können. Zumeist werden allerdings JSON- und XML-basierte Repräsentationen der mit Hilfe des APIs bereitgestellten Daten verwendet. Aus Sicht der Autoren kann die Identifikation, Spezifikation, Implementierung, aber auch das Management derartiger APIs, von den bei serviceorientierten Architekturen gewonnenen Erfahrungen profitieren. Während im europäischen Umfeld eher die Probleme des serviceorientierten Architekturansatzes diskutiert werden, finden sich in Nordamerika bereits kommerziell betriebene Verzeichnisse wie mashape, Xignite oder ProgrammableWeb, welche die Vorteile entsprechender Serviceangebote eindrucksvoll verdeutlichen.

2. Beiträge des Workshops

Im Folgenden findet sich eine kurze inhaltliche Zusammenfassung der auf dem Workshop gehaltenen Vorträge. Die korrespondierenden Artikel können im Tagungsband [Schmietendorf/Simon 2014] nachgelesen werden.

Harry M. Sneed, Stephan H. Sneed: SoA Integration als Alternative zur Code-Migration (eingeladener Beitrag)

> Die serviceorientierte Kapselung von Altsystemen, die in COBOL, PL/1, C, C++, C# oder auch in Java geschrieben wurden, steht im Mittelpunkt dieses Beitrags. Dabei wird auf notwendige Aufgabenstellungen und eine mögliche Werkzeugunterstützung eingegangen.

Frederik Kramer, Klaus Turowski: Auswahl und Parametrisierung einer Entscheidungsmethode zur Auswahl von Cloud Services in KMU

> Bei klein- und mittelständischen Unternehmen bleibt die Verwendung von Cloud Services aktuell noch hinter den Erwartungen zurück. Häufig liegen die Ursachen in einer ungenügenden Transparenz der Vor- und Nachteile. Für diese Entscheidungsfindung schlägt der Beitrag einen Ansatz vor.

Uta Pollmann, Frank Simon: Interoperabilität über Unternehmensgrenzen hinweg: Von SOA zum API-Management

> Die globale Interoperabilität ist für Themen wie IoT, M2M oder auch bei mobilen Apps essentiell. Dies geht mit einer SOA-fizierung der entsprechenden Schnittstellen einher. Im Einzelnen gehen die Autoren auf Fragen der Technologie, derSicherheit und des benötigten API-Managements ein.

Juraj Somorovsky, Markus Mayer, Mark O`Neill: SOAP to REST: Security Enhancement?

> Der Einsatz SOAP-basierter Web Services geht mit spezifischen Sicherheitsrisiken einher, welche im Beitrag erläutert werden. Weiterhin wird auf die Überführung von SOAP nach REST und den Möglichkeiten zur Gewährleistung der Sicherheit eingegangen.

Marco Mevius, Peter Wiedmann, Florian Kurz: Nutzerorientierte Multimedia-Geschäftsprozessmodelle als Basis der Serviceorchestrierung

> Dem Beitrag gemäß wird die Prozessmodellierung bei einer SOA zur Anforderungsanalyse, Servicedefinition und Serviceorchestrierung benötigt. Vorgeschlagen wird dafür die Verwendung der BPMN^{Easy}-Notation, deren Einsatz anhand eines Beispielszenarios verdeutlicht wird.

Michael Heydeck, Thomas Wiedemann: SOAlution – eine Praxislösung für das gruppenorientierte SOA-Praktikum

> Zur besseren Vermittlung der theoretischen und praktischen Kenntnisse, die im Zusammenhang mit dem Aufbau einer SOA benötigt werden, schlagen die Autoren eine werkzeuggestützte Lösung vor. Im Mittelpunkt des Systems stehen insbesondere die Funktionalitäten des Service Bus (vgl. ESB).

Jan Hentschel, Robert Neumann, Jörn Polifka, Joachim Wilken: Hadoop für „Big Processing“: Verteiltes Tile-Rendering zur Visualisierung der Fukushima-Radioaktivität in Japan unter Zuhilfenahme elastischer Cloud-Resourcen

> Der Beitrag zeigt, wie unter Zuhilfenahme elastischer Cloud-Resourcen, wie Microsoft HDInsight, die für das Rendering notwendigen Hadoop Cluster-Resourcen „on-demand“ zur Verfügung gestellt und nur für die Laufzeit der Berechnung in Anspruch genommen werden können.

André Nitze: Interoperability of Cross-Platform Mobile Services in Heterogeneous Environments

> Im Mittelpunkt des Beitrags stehen die Anforderungen für die Entwicklung mobil genutzter Geschäftsanwendungen. Dafür geht der Autor u.a. auf Fragen der Kosten, der Integration, der Qualität und der Sicherheit ein. Darüber hinaus wird ein webbasierter Entwicklungsansatz aufgezeigt.

Victor Czenter: Performancetesten in und aus der Cloud

> Im Mittelpunkt des Beitrags stehen Last- und Performancetests, die mit Hilfe von Cloud-Ressourcen ausgeführt, getrieben, konfiguriert und verwaltet werden. Unterschieden werden dafür die Einsatzszenarien System-under-Test, Test-Utility, Test-Umgebung und Test-Logistics.

Neben den aufgezeigten Vorträgen enthält der Tagungsband noch zwei Posterbeiträge. Diese beschäftigen sich mit dem Application Performance Management unter den Bedingungen von DevOps und serviceorientierten Schnittstellen bei NoSQL-Datenbanksystemen (speziell CouchDB).

3. Ergebnisse der Diskussionsrunde

3.1 *Bereitgestellte Diskussionsthemen*

Wie bei den vorangegangenen Workshops gab es abermals eine moderierte Diskussionsrunde zu aktuellen Trends, Herausforderungen und Hypes im Zusammenhang mit service- und cloudbasierten Architekturen. Zur Anregung der Diskussion wurden initial die folgenden Themen angeboten:

- API Management,
- Big Data und NoSQL Integration,
- Interoperabilität mobiler Services.

Die Moderation wurde durch Herrn Dr. Frank Simon (Head of Business Development - BLUECARAT AG) verantwortet.

3.2 *Ausgewählte Ergebnisse*

Die im Folgenden ausgewählten Diskussionsbeiträge wurden bewusst keiner Interpretation unterzogen, so dass sich darin auch gegensätzliche Meinungen wieder finden. Zur besseren Verständlichkeit erfolgte eine erste Strukturierung der verschieden aufgezeigten Themenbereiche.

Allgemeine Anmerkungen:
Im Sinne einer Selbstreflektion wurde der Sinn einer Interessensgemeinschaft im Zusammenhang mit serviceorientierten Architekturen kritisch hinterfragt. Immerhin existieren aktuell mehr als 200 Bücher zu serviceorientierten Architekturen. Darüber hinaus wurde eine zentrale SOA bereits als „tot“, die Idee der Serviceorientierung aber richtig charakterisiert. Provokant wurde die These einer zu starken Problemorientierung und eines zu geringen Lösungsbewusstseins in den Raum gestellt. Würde ggf. wird ein überarbeiteter SOA 2.0 Begriff benötigt? In diesem Zusammenhang stellen sich z.B. Fragen nach dem Zusammenwirken einer internen SOA mit extern benötigten (Cloud-) APIs, die Existenzberechtigung bzw. Sinnfälligkeit zentraler SOA-Komponenten oder auch die notwendige Prozessreife im Kontext erfolgreich eingesetzter SOA-Ansätze. Zunehmend geht es auch um die Berücksichtigung eines sich verändernden API-Begriffs, so dass zwischen einer quellcodespezifischen und einer geschäftsprozess- bzw. geschäftsobjektorientierten Sichtweise zu unterscheiden ist. Es gilt die Beziehungen zwischen intern genutzten Servicearchitekturen und extern verwendeten bzw. angebotenen APIs aus einer geschäftlich motivierten Sicht herauszuarbeiten. Dem entsprechend laufen automatisierte Geschäftsprozesse über Unternehmensgrenzen hinweg - die richtigen APIs ermöglichen diese Integrationsanforderungen. Darüber hinaus sollte sich die IT als Dienstleister verstehen, d.h. Fachabteilungen wollen nicht durch restriktive SOA-Vorgaben „gegängelt“ werden.

Aktuelle Herausforderungen:
Im zunehmenden Maße kommt es zu einem exponentiellen Wachstum benötigter Schnittstellen, d.h. „jeder spricht potentiell mit jedem“. Ein zentralistisch geführter SOA-Ansatz hat bei sich ständig verändernden IT-Infrastrukturen keine Chance. Selbst etablierte SOA-Lösungen können mit der geforderten Agilität einer zunehmend digitalisierten Welt nicht Schritt halten. Dem entsprechend werden leichtgewichtige Integrationsansätze benötigt, welche die folgenden Problembereiche lösen können:

- Bereitstellung eines Managements für heterogen eingesetzte APIs,
- Umgang mit proprietären APIs der verschiedenen Anbieter,
- Steuerung und Überwachung von Datenflüssen,
- Gewährleistung einer semantischen Interoperabilität,
- Berücksichtigung sicherheitstechnischer Anforderungen.

Interoperabilität wird vielfach als handwerkliches Problem betrachtet, so dass Probleme im Kontext von bilateralen Schnittstellenvereinbarungen gelöst werden. Aus technischer Sicht finden sich hier vielfältige Ansätze zur Etablierung von Integrationsarchitekturen wie z.B. CORBA, WS/EAI, WS/SOA, WS/REST oder auch JSON (WS – Web Service). Gerade in dieser Vielfalt von technologie- und produktzentrierten Lösungsansätzen liegt ggf. auch das Problem im Sinne einer nachhaltigen, wartbaren und erweiterbaren Verwendung. Darüber hinaus stellt sich die Frage, was eine SOA unter diesen Rahmenbedingungen überhaupt ist.

In reiferen Industriezweigen, wie dem Automobilbau, finden sich serviceorientierte Lösungsansätze, welche die Integration dutzender Steuergeräte innerhalb eines Automobils unter Verwendung des CAN-Busses erlauben. Auch in den eher konservativen Branchen, wie bei Banken und Versicherungen bzw. bei Telekommunikationsanbietern, wird die SOA-Idee weiterhin verfolgt. Aus diesen Ansätzen gilt es entsprechende Erfolgskriterien zu übernehmen.

Zusammenfassung

Durch die Teilnehmer des Workshops wurde die Diskussion in der folgenden Weise zusammengefasst:

- Die Systementwicklung ist „ex ante“ orientiert, d.h. sie wird durch ein „tagesaktuelles“ Denken ohne Berücksichtigung zukünftiger Herausforderungen bestimmt.

- Vorgehensweisen sind vom kurzfristigen Projektdenken geprägt, benötigte Lösungsansätze gilt es agil bereitzustellen.

- Zumeist entstehen Insellösungen, die ein lokales Optimum bezüglich der Interoperabilität bieten.

- Die Hoffnung liegt auf ggf. lernfähigen Schnittstellen, welche sich den Bedürfnissen entsprechend anpassen können.

Das Problem der Integration bleibt bei aktuellen Lösungen im Kontext von Industrie 4.0, Big Data oder auch mobilen Applikationen ein ständiger Begleiter. Zentralistische Lösungsansätze, die eine langwierige Standardisierung von Schnittstellen verfolgen, haben hier kaum eine Change. Das liegt auch an der Möglichkeit, sich über propritätere Schnittstellen vom Mitbewerber differenzieren zu können.

4 Weitere Informationen

Auch für das Jahr 2015 ist die Durchführung eines BSOA/BCloud-Workshops vorgesehen. Informationen zum Call for Paper für den kommenden Workshoptermin, finden sich unter folgender URL im Internet:

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

Alle Artikel des Workshops wurden innerhalb des 13. Bands der Schriftenreihe „Berliner Schriften zu modernen Integrationsarchitekturen“ beim Shaker-Verlag publiziert. (ISBN 978-3-8440-2940-6) [Schmietendorf/Simon 2014]

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Berliner Schriften zu modernen Integrationsarchitekturen
Hrsg.: Andreas Schmietendorf, HWR Berlin
Frank Simon, BLUECARAT AG Köln

Hochschule für Wirtschaft und Recht Berlin
Fachbereich II
Wirtschaftsinformatik – Systementwicklung

BSOA/BCloud 2014
9. Workshop Bewertungsaspekte
service- und cloudbasierter Architekturen

04. November 2014, Frankfurt/Main

SHAKER
VERLAG

Abbildung 2: Tagungsband zum BSOA-Workshop des Jahres 2014

5 Quellenverzeichnis

[Schmietendorf/Simon 2014] Schmietendorf, A.; Simon, F. (Hrsg.): BSOA/BCloud 2014 - 9. Workshop Bewertungsaspekte service- und cloudbasierter Architekturen (Frankfurt/M. - 04. November 2014), in Berliner Schriften zu modernen Integrationsarchitekuren, Shaker-Verlag, Aachen, November 2014

[Spencer 2015] Spencer, S.: The Service Oriented Business and how API's power the Service Oriented Startup, APIdays Sydney/Australia, February 2015, URL: http://syd.apidays.io/APIdays_program.pdf

1 Dank

Seit Gründung der BSOA-Initiative im Jahr 2006 erfährt diese vielfältige Unterstützung aus dem industriellen und akademischen Umfeld. Ein besonderer Dank geht an die Bluecarat AG als Gastgeber und Hauptsponsor der diesjährigen Veranstaltung. In diesem Zusammenhang sei auch Frau Marina Banduryanskaya, ebenfalls von der Bluecarat AG, für ihre umfängliche organisatorische Unterstützung gedankt. Ebenso sei der Ultra Tendency UG (Magdeburg) und der adhoc AG (Basel/Schweiz) für das Sponsoring gedankt. Organisatorische Unterstützung bei den vielfältig eingesetzten Websystemen zur Bewerbung der Veranstaltung erfuhr der Workshop von Herrn Dr. Dmytro Rud von der Roche Diagnostics AG/Schweiz, von Herrn Kevin Grützner und Herrn Stephan Hesseling von der HWR Berlin.

2 Organisation

Veranstaltet wurde der Workshop in Kooperation zwischen der Hochschule für Wirtschaft und Recht Berlin, dem Forschungszentrum Informatik Karlsruhe und der Otto-von-Guericke-Universität Magdeburg (Softwaremesslabor) unter der Schirmherrschaft der ceCMG (Central Europe Computer Measurement Group). Darüber hinaus erfährt die BSOA/BCloud-Initiative Unterstützung durch die GI (Gesellschaft für Informatik - Fachgruppe Softwaremessung- und Bewertung), die DASMA (Deutschsprachige Interessengruppe für Softwaremetrik und Aufwandsschätzung) und durch die ASQF (Arbeitskreis Software-Qualität und Fortbildung).

Quality-Based Issues in SOA Migration

Ayman Massoud

Otto-von-Guericke University Magdeburg, Germany

Abstract

Many organizations rely on complex enterprise legacy information systems to automate their business practices and collect, process, and analyze business data. These systems are large, heterogeneous, distributed, constantly evolving, dynamic, long-lived, and mission critical that presented as a backbone of the enterprise operations. To optimize business value, these large, complex systems must be modernized to new software paradigm like SOA "Service-Oriented Architecture". This migration process enables the organization to benefits from the new SOA capabilities, making the legacy functionalities more robust, efficient and cost effective to align easily with the new business opportunities.

Several migration frameworks are presented to facilitate and manage the migration activities, most of these frameworks considered deeply technical analysis of understanding the legacy system and the transition steps to the target system. However, considering the efficiency and quality requirements and measurements throughout the migration tasks and activities are still needs more research contributions, in order to avoid the repeating of the legacy limitations in the new environment, and to produce more reliable, integrity, and efficient SOA solution.

This paper is divided into five parts; the first one is an introduction to explain the key research motivation and objective. Second part illustrates the related work of the SOA migration approaches, architectures, frameworks, and methods to understand the quality and evaluation measurement challenges and issues in SOA migration. The third part executes comparison and gap analysis exercise between the presented migration frameworks from the quality and measurements perspectives. The forth part explains our quality proposal model that we presented to support the migration quality issue. And finally in the fifth part, the paper presented in brief a new SOA migration framework SMF that adopted the proposed quality requirements model and the E4 measurement approach in order to execute a new quality-based SOA migration process.

1 INTRODUCTION

Modernize the mission-critical legacy systems is supported the organization to transfer its outdated systems into new software paradigm that making these applications more robust, agile enough to align easily with the new business opportunities, more efficient and cost effective. Despite of the fact that the SOA migration process is succeeded to make the legacy systems running and benefits from the modern target architecture, some of the legacy limitations and issues are still exists, and some of the migration outcomes are not efficient as expected. Accordingly and as per our literature survey; there are several approaches to migrate legacy applications to SOA have been reported in academic and in industry. Out of our publications survey, we have summarized some related papers that reporting legacy to SOA migration during the period from 1997 to 2015.

Service-oriented approach is the most significant software modernization reported in the current software engineering domain, presented as a solution to overcome the legacy systems limitations and issues. The objective of this paper is to present new qualified-based SOA migration approach that discusses how to design, implement, and evaluate an efficient SOA migration framework with acceptable level of migration quality that produces reliable, efficient, and consistent results.

2 SOA MIGRATION FRAMEWORKS – LIETRATUR SURVEY

Based on the publications on the software re-engineering domain (high number of citations, and the availability of documentations); the following methods are used to identify the SOA migration phase's framework:

- Butterfly Method (Wu, et al., 1997)
- Renaissance Method (Warren & Ransom, 2002)
- Architecture-Driven Modernization - ADM (Khusidman & Ulrich, 2007)
- IBM's SOMA Method (Arsanjani et al. 2008, Fuhr et al. 2011)
- Service Migration and Reuse Technique - SMART (Lewis 2005, SEI 2008)
- SOA Migration Framework SOA-MF (Razavian and Lago**,** 2010)
- SOA Migration - SOAMIG [Zillmann et al., 2011]
- Consolidation framework of structural legacy to SOA Migration [Khadka, et al., 2013]
- Advanced Software based-service provisioning and migration of legacy Software [ARTIST Project, 2015]

In the following subsections, we will explain in more details the selected literature of the SOA migration methods, approaches, and frameworks:

2.1 Butterfly Method

Butterfly is an approach to mission-critical legacy system migration: the Butterfly Methodology, its data migration engine and supporting tool-kit framework. Data migration is the primary focus of the Butterfly methodology; however, it is placed in the overall context of a complete legacy system migration. Butterfly method is consists of 5 migration stages; namely: justification, legacy system understanding, target system understanding, migration, and testing. The methodology is depicted in the following figure 1:

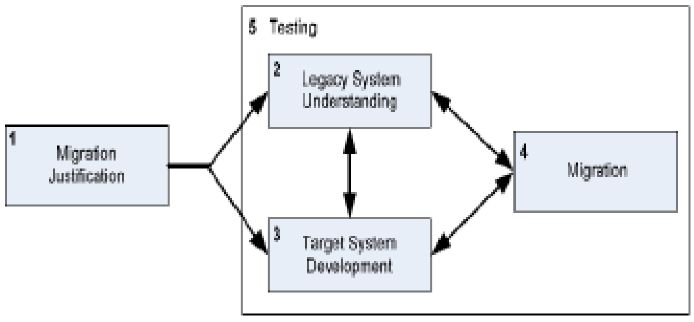

5 Testing
2 Legacy System Understanding
1 Migration Justification
4 Migration
3 Target System Development

Figure 1: Butterfly method (Wu, et al., 1997)

Justification phase explains the risk and the benefits associated with the legacy system modernization, based on which the decision of modernization or re-development has to be taken. To support such decisions, various activities are carried out, for instance, cost benefit analysis to determine the economic benefits, software quality metrics to determine the technical feasibility.

Legacy system understanding uses the reverse engineering method to identify the legacy components, recreate documentation, understand the static and dynamic behavior of the legacy system, and create the representations of the system at a high level of abstraction.

Target system development classifies the requirements/specifications of the target system and choosing the most appropriate architecture and standards to support the goals that specified in the legacy system understanding phase.
Migration phase is concerned with the physical movement of the whole legacy system to the target system.

Finally, Testing is carried out throughout the modernization process to ensure that the target system delivers the functionalities specified at the starting of the modernization.

Butterfly Method Characteristics: The objective of the Butterfly methodology is to migrate a mission-critical legacy system to a target system. The fundamental premise of the Butterfly methodology is to question the need for parallel operation of the legacy and target systems during migration. The Butterfly methodology eliminates, during the migration, the need for system users to simultaneously access both the legacy and target systems, and therefore, eliminates the need of interoperation between heterogeneous information systems.

Butterfly methodology is focusing on the data migration, therefore it is proposes a legacy data migration engine, suitable for mission-critical system migration. Butterfly method is not considered how to evaluate or measure the migration quality and process efficiency, it is just mentioned that the important aspect of migration testing is to ensure that there are no unexpected inconsistencies between the critical functionality of the legacy system and its replacement.

2.2 Renaissance Method

The renaissance method for legacy system modernization consists of 4 phases, namely; plan evolution, implement, deliver, and deploy & use. Each phase is further categorized into key activities. The renaissance method is displayed in figure 2:

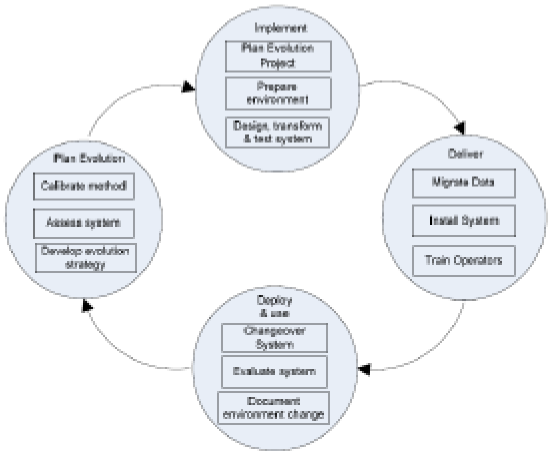


Figure 2: Renaissance method (Warren & Ransom, 2002)

Plan evolution phase involves three sub elements which addresses the system's long-term future; Calibrate method activity that involves gathering information and feedback from organizational units to assess the need of evolution. The assess system activity involves the assessment of the legacy system from economical, technical and business organizational perspective. Upon assessing the legacy system, proper evolution strategy is developed as a last activity is this phase.

Implement phase in this phase the modernization project determines which evolution strategy to implement for evolution, prepare environment that determines the requirements of the target system and selecting the appropriate standards and technologies for the target system. The design, transform and test system activity involves the implementation of the evolution and testing the implementation technique.

Deliver phase including; migrate the legacy data into the new system, install the transformed system after evolution and train operators on the new migrated system.

Finally, deploy & use phase is concerned with the deployment of the transformed system. This phase includes cutover plant for gradually stopping the operation of the legacy system and using the new migrated system, determine the effectiveness of the evolution and create new documentation in the course of evolution.

Renaissance Method Characteristics: Renaissance supports system evolution by first recovering a stable basis using reengineering, and subsequently continuously improving the system by a stream of incremental changes. Renaissance method can be tailored to the needs of particular projects and organizations, and it is not prescriptive of particular tools and techniques. The objective of Renaissance is providing a controlled approach to system change essentially means reducing the costs and risks associated with change. The Renaissance method comprises a classification of evolution strategies, a process framework, an information repository, and a set of responsibilities to be met in a typical evolution project. Each of these elements can be tailored to fit particular project and organizational factors. The Renaissance method is determined four requirements that shape its characteristics:

R1 Method should support incremental evolution.

R2 Where appropriate, method should emphasized reengineering, rather than replacement.

R3 Method should prevent the legacy phenomena from reoccurring.

R4 It should be possible to customize the method to particular organizations and projects.

And used these requirements to evaluate and measure its strength and weakness as follows:

- **Strengths**: Well-defined process, Application assessment method, Evolution strategy selection process, Customizability, Protection of investment in current systems, and Business-driven nature.
- **Weakness**: Adoption overhead, and Overhead for small projects.

2.3 Architecture-Driven Modernization Method (ADM)

The architecture-driven modernization method is based on the reengineering horseshoe model (Bergey, et al., 1999). The ADM horseshoe model (Figure 3) consists of three major architectural perspectives namely: business architecture, application and data architecture and technical architecture. Left side of the Figure 3 represents the existing legacy system and similarly the target system in the right with its three levels of architectural perspectives. The curve from legacy to target system represents the transformation path of modernization.

The ADM involves transforming the existing legacy system incrementally to the target system in any architectural perspective. For instance, the evolution can be in technical architectural level that involves the transformation of legacy code to object-oriented code. As per the ADM any transformation curve representing the modernization has three elements: knowledge discovery of the legacy system, target architecture definition and transformative steps.

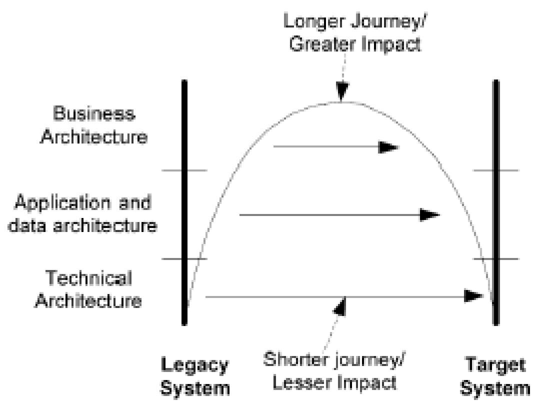


Figure 3: ADM horseshoe model (Khusidman and Ulrich, 2007)

The knowledge discovery of the legacy element involves the reengineering of the legacy system to understand it. The target architecture definition element determines the target solution/architecture and its details into which the legacy code can be mapped or transformed. Finally, the transformative steps of ADM including migrate the legacy system to the target system. The transformation can be at any abstraction level ranging from the physical code level (e.g. language migration) to a more abstract level (e.g. business rule transformation).

ADM Method Characteristics: ADM method considered that the modernization is summarized in three perspectives based on the architectural domains those projects impact; these perspectives (transformation phases) are business architecture, application and data architectures, and technical architecture. The ADM path represents the way knowledge from the existing solution is discovered, enhanced and reused in the target solution. ADM method adopted three elements to every transformational path, regardless of the level of architectural impact:

1. Knowledge discovery of the existing solution. This can occur at many levels of abstraction across varying degrees of scope as appropriate to the projects involved.

2. Target architecture definition. In order to create a transformation approach, analysts must create a target solution that serves as a framework into which existing solutions can be mapped or transformed.

3. Transformative steps that move the as-is state to the to-be state. The approach can range from the physical (e.g. a language migration) to the more abstract (e.g. business rule mapping to a rules-based environment).

The ADM method is referred to (without more details of applied techniques) the software assurance and metrics as standards to be used in the modernization processes.

2.4 Service-Oriented Modeling and Architecture (SOMA)

Service-Oriented Modeling and Architecture SOMA developed by IBM [Arsanjani et al 2008], SOMA is an iterative and incremental method to design and implement service-oriented systems. SOMA describes how to plan, design, implement, and deploy SOA systems. SOMA is designed extensible to be able to include additional, specialized techniques supporting specific project needs. In the following figure 4 [Fuhr, 2011] the seven SOMA phases are illustrated as follows:

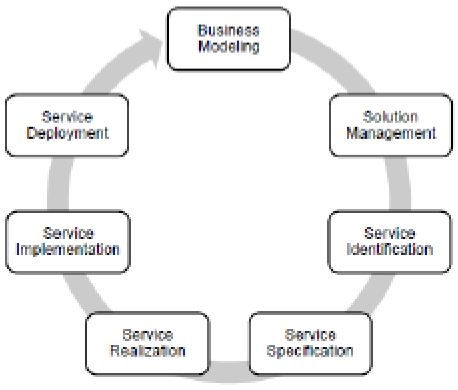


Figure 4: The seven SOMA phases (Arsanjani et al 2008)

Regarding the development and implementation of SOA, several methods are available: SOMA (Arsanjani et. al., 2008), TOGAF (The Open Group Architecture Framework, 2007), MDA (Model Driven Architecture Truyen, 2006). While TOGAF and SOMA could be applied for broader context as enterprise, Model Driven Architecture is used mainly within a project scope. Another particularity of MDA is that it is focused on models with different degrees of abstractions (Computation Independent Model, Platform Independent Model, and Platform Specific Model) instead of phases. TOGAF and SOMA are formulated as multiple-phase methods that are executed incrementally. However, there is a difference between them: SOMA phases identify the main processes of SOA development, while TOGAF phases relate to various domains where the focus should be oriented during SOA development such as Business Architecture, Technology Architecture, and Migration Planning.

Business Modeling, the state of a company is analyzed at the beginning of a project. As SOAs are tightly aligned to business concerns, it is necessary to clearly understand the customer's business. In this phase, all possible information about the following concerns is gathered:

– Business mission and vision

– Business actors, use cases and processes

– Business challenges

– Business goals and key performance indicators (KPIs)

One main result of this phase is the business model which is a formalized view on these aspects.

Solution Management adapts the SOMA method to the project needs. This includes choosing additional techniques to solve project-specific problems. From a SOMA perspective, the SOA project is located in Solution Management since it adapts SOMA to software migration issues, using model driven technologies.

During **Service Identification**, SOMA uses three complementary techniques to identify service candidates, i.e. functionality that may be implemented as service later in the new SOA architecture. Domain Decomposition is a top-down method decomposing the business domain into functional areas and analyzing the business processes to identify service candidates. Goal-Service Modeling identifies service candidates by exploring the business goals and sub goals. Legacy Asset Analysis finally explores the functionality of legacy systems bottom-up. It is analyzed, which business processes are supported by what functionality of a legacy system. For that purpose, documentation, APIs or interfaces are explored to identify which functionality is provided. The source code is only analyzed on a coarse-grained level, meaning it is analyzed which functionality exists and not how it is actually implemented. For each business function that supports the business process, a service candidate is created. All three techniques are performed incrementally and iteratively. For each identified candidate, an initial service specification is created and a trace to the source of identification is established.

Service Specification deals with describing the service design in detail. The initial service specification is refined, messages and message flows are designed and services are composed. This phase results in a comprehensive description of the service design. SOMA uses an UML profile for Service-Oriented Architectures to describe the service design. Later, the specification will be transformed into WSDL code for implementing the service as a Web Service (as it is proposed by SOMA and it is much common used).

Service Realization decides which services will be implemented in the current iteration and constitutes how to implement them. First, identify service candidates that should be exposed using a set of criteria to evaluate usefulness and value of each service. After having chosen a set of services, the implementation strategy has to be defined. Encapsulation of services allows the choice of different ways to implement each service. Common strategies to form new service components are:

1. Implementation from scratch,

2. Wrapping of larger legacy components or

3. Transforming the required legacy components.

After having decided on an implementation technique, legacy systems require fine-grained analysis. Functionality that is able to implement services has to be identified in the legacy code. In addition, it is important to clearly understand how this functionality is embedded in the legacy system, since it has to be separated to build a self-contained service. Finally, the implementation design specifies how to implement the service.

During the **Service Implementation** phase, services are actually implemented. According to the decisions derived in the Service Realization phase, services are developed, wrappers are written, or legacy code is transformed. Finally, all services are orchestrated and message flows are designed. The last phase is **Service Deployment**. It deals with exposing the services to the customer's environment. Final user-acceptance tests are performed and the SOA is monitored to verify that it performs as expected.

SOMA Method Characteristics: SOMA defines key techniques and describes the roles on a SOA project and a work breakdown structure (WBS). The WBS includes tasks, the input and output work products for tasks, and the prescriptive guidance needed for detailed analysis, design, implementation, and deployment of services, components, and flows needed to build a robust and reusable SOA environment. SOMA methods includes seven migration phases; Business Modeling, Solution Management, Service Identification, Service Specification, Service Realization, Service Implementation, and Service Deployment. SOMA phases are not linear. They are applied in a risk-driven, iterative, and incremental approach using a nuance peculiar to the SOA life cycle.

SOMA method is focus on packaging, provisioning, executing user-acceptance testing, and deployment of services in the production environment. In addition, SOMA provides support of monitoring and management of business processes and performance monitoring in the production environment. SOMA also provides linkages to runtime monitoring and management aspects, as in system, infrastructure, and network management.

2.5 Service Migration and Reuse Technique (SMART)

Service Migration and Reuse Technique (SMART) [Lewis et al., 2008] is an approach for making decisions on the migration of legacy components to services. SMART helps organizations to make initial decisions about the feasibility of reusing legacy components as services within an SOA environment. SMART was initially developed in 2005 [Lewis et al., 2005 and 2006].

SMART consists of four elements:

1. The SMART Process is a systematic means to gather information about the legacy components, the candidate services, and the target SOA environment.

2. The Service Migration Interview Guide (SMIG), guides the discussions during the initial SMART process activities. It contains more than 60 categories of questions that gather information about the migration context, the legacy components, the candidate services, and the target SOA environment. The goal of using the SMIG is to assure broad and consistent coverage of the factors that influence the cost, effort, and risk in migration to services. Each question in the SMIG is associated with potential migration issues or aspects that are known to require extra cost or effort.

3. Using the SMIG as a framework, the SMART Tool automates data collection and relates answers to questions to potential risks to mitigation strategies. Then, answers and associated information yield a draft migration strategy and migration issues list. The tool also consolidates data from multiple engagements for trend analysis.

4. Artifact Templates for output products are created as part of the process. These templates, which are initially populated by the SMART Tool, include the following: 11

   - **Stakeholder List:** Contains the information about all stakeholders who will provide input into the process—sponsors, managers, system developers, system maintainers, system architects, representatives of service consumers, and IT staff.

   - **Characteristics List:** Contains the list of characteristics that needs to be gathered about each component targeted for migration. It initially contains basic information such as name, function, size, language, operating platform, age and gets updated as migration issues are identified.

   - **Migration Issues List:** Contains the list of migration issues that are identified during the information-gathering activities.

   - **Business Process-Service Mapping:** Contains the mapping between main business processes and candidate services.

- **Service Table:** Contains information about candidate services such as description, associated legacy components, inputs, and outputs.

- **Component Table:** Contains information about legacy components targeted for migration as identified in the Characteristics List.

- **Notional Service-Oriented System Architecture:** Presents a high-level view of the system architecture showing service consumers, infrastructure components, services, and legacy components, as well as their interaction.

- **Service-Component Alternatives:** Presents the different options for satisfying candidate service requirements. Options are wrap, extract, create new, rewrite in a different language, add external service, acquire commercial product, or fashion any combination of the above.

- **Migration Strategy:** Contains the migration strategy for the targeted legacy components, as well as guidance for future migration efforts.

The following sub-sections outline the SMART elements process as shown in figure 5.

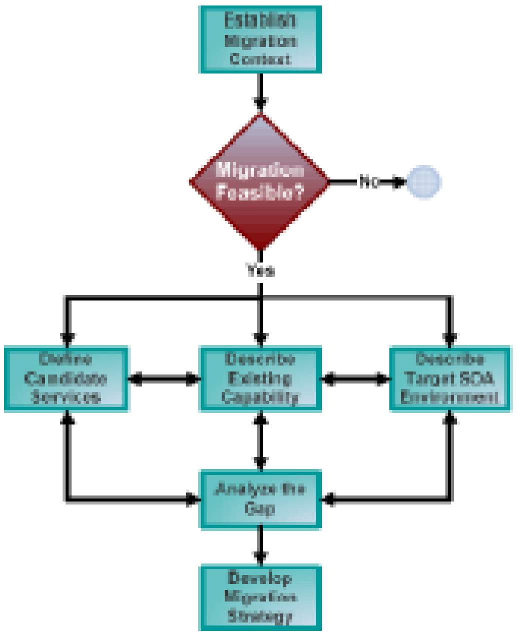

Establish Migration Context
Migration Feasible?
No
Yes
Define Candidate Services
Describe Existing Capability
Describe Target SOA Environment
Analyze the Gap
Develop Migration Strategy

Figure 5: The SMART Process (Lewis et al., 2008)

The SMART Process has six activities and one decision making, establish context, feasibility decision, define candidate services, describe existing capabilities, describe the target SOA environment, analyze the Gap, and finally develop strategy.

Establish Context: The *Establish Context* activity has the following tasks:

- Understanding the business and the technical context of the migration project, including organization business and technical objectives, target SOA expectations, project stakeholders and time schedule, project budget and constraints, and any other relative topics that support to understand the migration context.

- Understanding the legacy and the SOA systems, its functionalities, technologies, limitations, benefits, etc…

- Identify a set of candidate services for migration. Using top-down and bottom-up approaches based on the migration drivers.

During the *Establish Context* activity, the following artifacts are initially developed:

Stakeholder list, migration issues list, characteristics list, and business process-service mapping

Migration Feasibility: The process of the feasibility study is essential to determine if the legacy systems are potential enough to be represented as services or not. The case is varying between the following scenarios:

Feasible

- Migration goals are clear and valuable for all stakeholders.
- Both legacy and target systems are well understanding.
- Candidate services and its consumers are identified.
- Initial mapping of services to legacy component has been done.

Not Feasible

- Services consumers are not identified.
- Build services from legacy code are not potential for common use.
- There appears to be incompatibility between the legacy and the target SOA environment.
- No stateless functionality in the legacy system.

Feasible but need additional information

- Clarify the business goals that expected from the migration project.
- Services consumers should to be well identified.
- Availability of key stakeholders to suppo rt the migration project.
- Identification of target SOA environment

Candidate Services Identification: In this activity the identification of the potential candidate services is the main goal, the selection is based on the most services that has clear input and output, perform concrete function, can be reused across several applications, support the QoS requirements.

Describe Current Capabilities: The goal of this activity is to discover and understand the legacy capabilities and limitations toward service-oriented solution. This activity may include:

- Descriptive data about legacy components, its name, function, size, language, operating platform, age.
- Architecture views, design paradigms, system quality, change history, user satisfaction, and existing problems.

Additional information needed about components will be determined by the migration issues that emerge during the process.

Describe Target SOA System: This activity gathers information about the target system (SOA environment) for the selected services including

- Major components of the SOA solution
- Impact of specific technologies and standards used in the migration
- Guidelines for service implementation
- State of SOA system
- Interaction patterns between services and the overall solution
- QoS expectations and execution environment for services

Gap Analysis: This activity is focusing in calculates the cost, time, risk, and effort estimated to complete the migration process, given the candidate service requirements and target SOA characteristics. The discussion of the changes that are necessary for each component is used as the input to calculate this preliminary estimation.

In some cases, additional analysis methods may be needed, such as evaluation of code quality using code analysis tools or architecture reconstruction. For example, if the dependencies between components of the system are not well known and the technical personnel is not capable of providing details of the changes or the magnitude of the changes, an architectural reconstruction could provide a set of views to understand these dependencies [Kazman 2002, O'Brien 2002].

Develop Strategy: Develop strategy activity is aim to draw the road map of migration processes that taking in considerations all the output coming from the previous activities. This activity may include:

- Feasibility, risk, and options for proceeding with the migration effort
- Starting with pilot project to see how the migration proceed using samples of candidate services and legacy components
- Guidelines and options that suppo rt all the migration implementations tasks
- Issues to be addressed and recommendations for mitigations common problems.

SMART Method Characteristics: Service Migration and Reuse Technique (SMART) is a migration method that considered concrete analysis of the feasibility, risk, and cost involved. SMART process helps organizations to make initial decisions about the feasibility of reusing legacy components as services within an SOA environment. SMART gathers information about legacy components, the target SOA environment, and candidate services to produce (1) a preliminary analysis of the viability of migrating legacy components to services, (2) an analysis of the migration strategies available, and (3) preliminary estimates of the costs and risks involved in the migration.

2.6 SOA Migration Framework (SOA-MF)

According to [Razavian and Lago, 2010], the SOA Migration process is considered as some kind of reengineering process as in [Kazman, et al., 1998], including reverse engineering, transformation, and forward engineering process, and that the horseshoe model is a generally accepted conceptual model for reengineering. SOA Migration Framework SOA-MF is a proposed framework that extended form of the horseshoe model as a holistic model of the migration process. Figure 6 of SOA-MF illustrated the migration process phases: (Reverse Engineering, Transformation, and Forward Engineering)

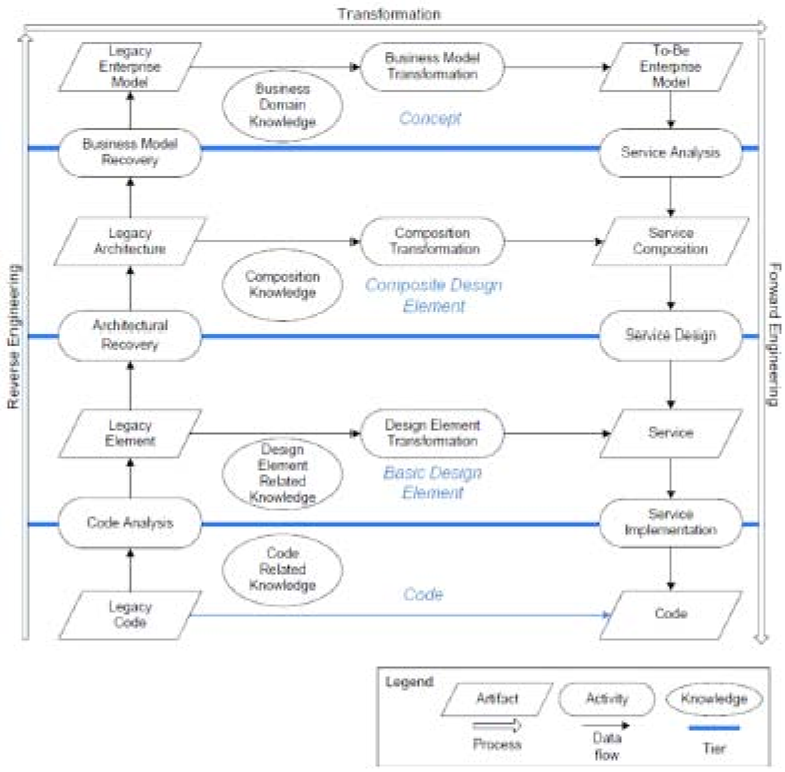


Figure 6: SOA Migration Framework SOA-FM (Razavian and Lago, 2010)

The mentioned framework illustrates the migration process together with details of the artifacts included, activities carried out and types of knowledge exploited within each of migration sub-processes. The graphical representations of the conceptual elements are depicted in Figure 2.5. The sub-processes, activities, artifacts and knowledge elements are respectively depicted by thick arrows, rounded rectangles and parallelograms.

2.6.1 Reverse Engineering

Reverse engineering sub process starts from analyzing the legacy code within the code analysis activity. The input artifact of this activity is the legacy code while the output consists of set of legacy elements (which could be in the form of components, modules, segments of code, etc.). The extraction of legacy elements from code is influenced by involvement of code related knowledge (such as code grammar and model) as well as bodies of knowledge addressing higher level concepts (such as business domain knowledge). Within the reverse engineering sub process, the extracted legacy elements are inherently design entities recaptured by means of reverse engineering techniques. However, SOA-FM is go one step further and recaptures the meaningful compositions of these legacy elements within the architectural recovery activity. In this phase, the composition knowledge such as architectural patterns and architectural styles are involved in identification of the architectural elements and their associated relationships.

Finally, the legacy enterprise model is extracted during the business model recovery activity as output, while the inputs to this activity are the legacy architecture as well as the existing business domain knowledge such as business rules, business processes, etc.

2.6.2- Transformation

Transformation is restructuring one representation form to another at the same level of abstraction, transformation process in the SOA-MF is encompasses 3 main activities:

- **Design element transformation** activity is typically performed to move the encapsulation of the legacy elements (extracted during the reverse engineering process) to services. Most of the wrapping techniques fall in this category of transformations. The input artifact to this activity is the legacy element (i.e. module, component or segment of a code) while the output artifact is basically a service.

- **Composition transformation** this activity transform the legacy architecture to service compositions (components and connectors are transformed to a service composition embracing services and relationships among them).

- **Business model transformation** in this activity the existing business model is transformed to a to-be business model based on new requirements as well as opportunities offered by service based systems. Here, existing business rules, business processes and strategies which are partially embedded in the legacy enterprise model are transformed to new ones to form the basis for development of service based system. The input artifact to this activity is legacy enterprise model, whereas the to-be enterprise model is the output. The business model transformation activity is assisted by the business domain knowledge such as business rules, risks, benefits and plans.

2.6.3 Forward Engineering

The output of the previous migration process is “To-Be- Enterprise Model”, this model is produced migrated services throughout:

- **Service Analysis** During service analysis, based on the to-be enterprise model a set of candidate service compositions which conceptualize the business processes are identified.

- **Service Design** renovated services are designed based on the consolidated candidate service compositions.

- **Service Implementation** candidate services are merged with the services identified during design element transformation activity. Finally, during service implementation the service design is transformed to code.

SOA-MF Method Characteristics: SOA migration framework (SOA-MF) characterize and isolate the properties of migration approaches in terms of processes it supports, artifacts included, activities carried out, and types of knowledge exploited. SOA-MF considered that the notion of tier plays an important role in positioning and classifying the various migration approaches. The tiers of SOA-MF covered by a specific SOA migration approach can explain the following aspects: the associated level of abstraction in which the transformation occurs and the transformations that entail lower level ones.

2.7 Model-Driven SOA Migration (SOAMIG)

SOAMIG is extend of IBM's SOMA method (Service-Oriented Modeling and Architecture [Arsanjani, et al., 2008]) [Zillmann, et al., 2011], [Fuhr, et al., 2011], can be viewed as an extension of SOMA using "graph-based reverse engineering and transformation techniques to enable model-driven software migration" (Fuhr, Horn, & Winter, 2010).

SOAMIG's characteristics position it in the field of legacy system migration. SOA migration planning is dealt with, in the SMART approach (Smith, 2007) also. A graph based migration approach has been proposed by Matos (2008) as well. Furthermore Correia et al. (2007) and Fleurey et al. (2007) have described approaches of model-driven migration. Lastly, a legacy system migration procedure with wrapping as its strategy's focus has been proposed by Marchetto and Ricca (2008) as well as Gimnich (2007).

SOAMIG aims at defining an adaptable iterative migration process model. The SOAMIG process distinguishes four organizational phases exposing important milestones in migration projects (Figure 7-A). The phases included several disciplines of activities during migration:

1- **Preparation:** This phase is starting from the legacy code which has to be prepared and standardized in the Pre-Renovation discipline by various reengineering activities to alleviate conversion activities. The migration project infrastructure including defining project goals and work packages or managing resources is set up in the Project Setup discipline. Migration projects require a high level of automation by using appropriate tools. General development of reengineering and conversion tools is covered by Tool Initialization; their adaptation to detailed project-specific requirements is addressed in Tool Adaptation in the next Conceptualization phase.

2- **Conceptualization:** A central activity in migration projects is assessing feasibility of migration and applicability of provided tool sets during Technical Feasibility.

3- **Migration:** Migrated the entire system is applied after setting up a general migration strategy and tool support. In the Migration phase, all SOAMIG core disciplines are performed iteratively in different intensities, resulting in a migrated system in production.

4- **Transition:** Code migration usually leads to hardly maintainable code, which requires additional reengineering. Software quality degrades by adopting mindsets from legacy to target structures directly [Teppe, 2009]. The quality of the migrated system has to be improved in the Post- Renovation discipline in the target environment.

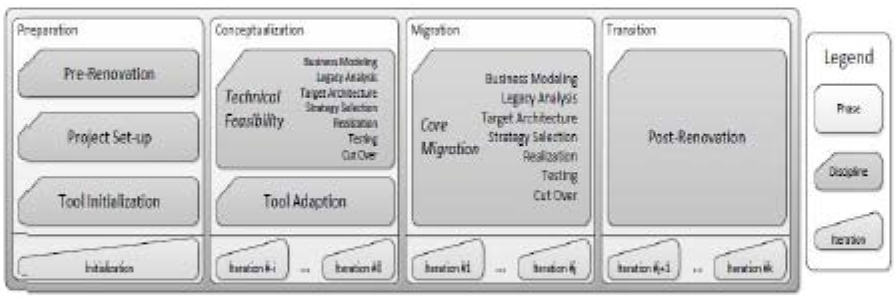


Figure 7-A: SOAMIG - SOA Migration Framework (Zillmann, et al., 2011)

The SOAMIG Core Disciplines: The SOAMIG Core disciplines (Figure 7-B) are performed during Conceptualization phase for a small part of the legacy system and eventually in the Migration phase for the entire system. Most of these disciplines use model driven techniques based on an integrated repository [Fuhr, 2010], [Zimmermann, 2010].

1. **Business Modeling:** the objective of SOAMIG is the migration to SOA, which requires analyzing the business processes of legacy systems to allow a reasonable tailoring of services in the Target Architecture discipline. The evaluation and documentation of supported business processes is handled by the Business Modeling discipline using UML2 activity diagrams and Business Process Modeling Notation (BPMN). These models are integrated with architecture and code models in the SOAMIG repository.

2. **Legacy Analysis:** Legacy Analysis deals with exploring and comprehending the legacy system. Available information like user or technical documentation, test cases, architecture description and source code have to be analyzed. In SOAMIG, static and dynamic analysis techniques including FGM (Flow Graph Manipulator) [Beier, et al., 2009] and JGraLab/GReQL ([Ebert, et al., 2008], [Ebert, et al., 2010]) are applied. Service candidates are discovered by mapping business processes from Business Modeling to the legacy.

3. **Target Architecture:** Finding a best target architecture deals with both, the legacy system and the required software support [Zillmann, et al., 2010] in the target system. The target architecture is iteratively approximated, starting from a technically ideal architecture and taking into account special requirements of the legacy to enable economic migration. The SOA target architecture consists of service design, the realization design and the orchestration design. The service design describes the interfaces of the target architecture services. The realization design describes how to implement the services or the user interfaces, and finally the orchestration design specifies how to orchestrate services to support business processes.

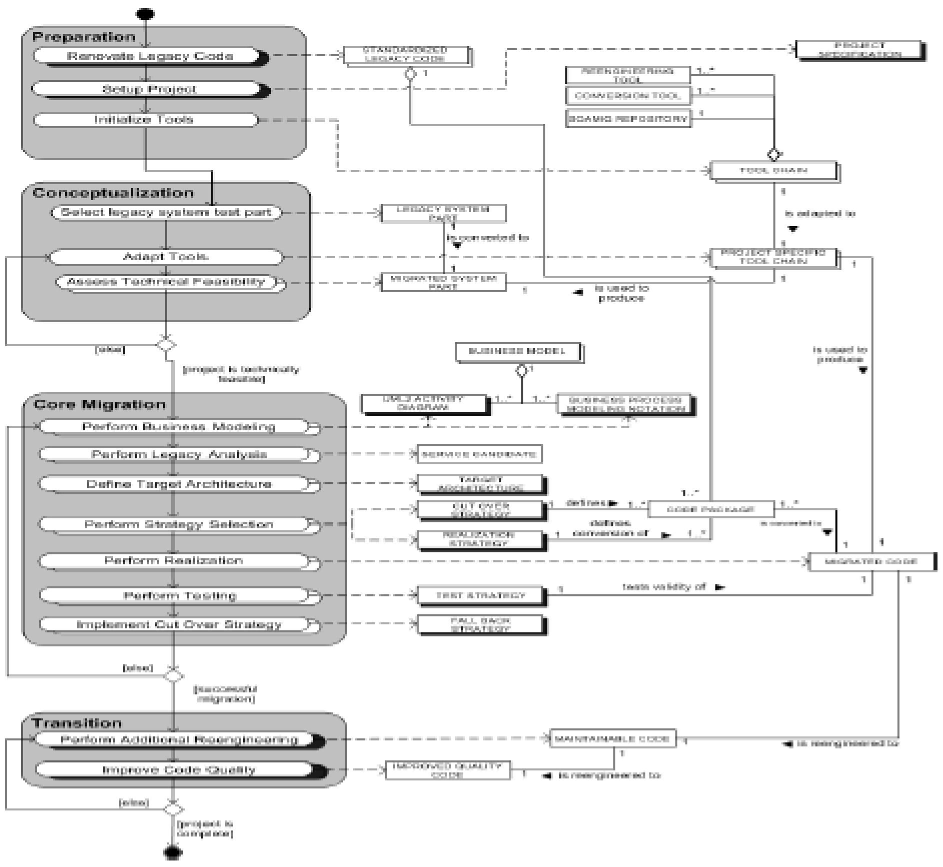


Figure 7-B: SOAMIG - Core Migration Activities (Zillmann, et al., 2011)

4. **Strategy Selection:** Strategy Selection decides on the cut-over strategy, which defines delivery of (parts of) the migrated system and on the realization strategy for converting each package. Cut-over strategies vary from conversion in one go (big bang) to iterative strategies, providing stepwise migration and calling for bridging architectures to enable collaboration of parts of legacy and target system [Brodie, et al., 1995]. Performing iterative migrations also includes deciding on the parts of the system to be migrated in each iteration. The realization strategy addresses the conversion of each migration package. This includes project, package and service realization strategies. Alternative strategies are reimplementation, transformation-based conversion, and wrapping. The corresponding strategy is selected according quality and business value of each migration package [Bennett, et al., 1999].

5. Realization: in this discipline, functionality of the legacy system is converted to the target system. Migration projects deal with migrating functionality, user interfaces and data, etc. SOAMIG especially focuses on transformation-based migration. So, it is aspired to convert as much code as possible by an automated transformation using SOAMIG converters and translators. In SOA migrations, a special focus lies on services and service orchestration. Whereas service functionality could be extracted and migrated (semi-)automatically, the orchestration of services usually has to be newly implemented since legacy systems lack the required orchestration information.

6. **Testing:** Testing deals with ensuring equivalent behavior of legacy and migrated system by applying regression tests from the legacy system to the migrated system. System tests account for correctness within the target environment. The chosen testing strategies depend on the embedding of the migrated system.

7. Cut Over: Cut Over concludes the core migration in SOAMIG. The migrated system is deployed at the customer’s site, while the legacy system is turned off. To keep decisions and results based on the legacy system comprehensible for future analysis, in some cases, the legacy has to be preserved. Cut Over follows the cut-over strategy selected in Strategy Selection. A fallback strategy is required to ensure switching back to the old system without loss, if serious errors occur during migration. This also includes a reverse migration procedure to reconvert e.g. data changes already made in the target system before fallback [Teppe, 2009].

SOAMIG Method Characteristics: The SOAMIG process is divided into four distinct phases, each being a milestone during a migration process. Each phase incorporates several disciplines while, at the same time, a set of core disciplines is presented that influences the two main phases of the procedure. Lastly, besides the first, all phases pass through a multitude of iterations before their completion.

SOAMIG Core disciplines are performed during Conceptualization phase for a small part of the legacy system and eventually in the Migration phase for the entire system. Most of these disciplines use model driven techniques based on an integrated repository. SOAMIG considered testing phase to ensure equivalent behavior of legacy and migrated system by applying regression tests from the legacy system to the migrated system.

2.8 Consolidation Framework of SOA Migration

According to [Khadka and et al. 2013], [Kontogiannis et al., 2008], [Lewis et al., 2008], [Lewis et al., 2009], and [Lucia et al., 2008] of SOA migration evaluation process, several approaches to migrate legacy applications to SOA have been reported. Some approaches are proposed in academia [Khadka and et al. 2012], [Razavian and Lago, 2010] and others are proposed in industry [Razavian and Lago, 2011], [Razavian and Lago, 2012]. These approaches can be basically categorized into two aspects: **migration planning** to determine the migration feasibility based on technological and economical assessments, and **migration execution** to develop a supporting technology so as to expose legacy applications as a service and to provide service provisioning upon exposing the service.

These given approaches are considered that SOA migration process requires the consolidation of both planning and execution migration aspects. And considered also that the legacy to SOA migration is not only a complex technical endeavor, but it also involves various organizational and business perspectives [Nasr et al., 2011]. The mentioned researchers are classified the process of SOA migration framework and its structure in six phases as a complete merged scenario (figure 8-A [Khadka et al., 2013]), started with planning stage to understand the legacy and the SOA systems requirements, and then provide the study of migration feasibility from the technical and ecumenical perspectives according to the given context. So, upon completed this planning stage the decision can be taken to move to the implementation stage which involves; identified the proper services to be created and re-used, select or create a technique to expose/leverage the legacy functionalities to a services, and finally the implementation stage manage the legacy-services deployment and the provisioning to start the migration go-live process

The following sub-sections will illustrate these migration phases in more common details:

2.8.1- Legacy System Understanding (LSU)

Understanding the legacy system and it's as-is situation are crucial to the success of any evolution [Seacord and et al, 2003]. LSU target to understanding what the legacy system do and how it can do it, via deeply analysis in legacy system for acquiring information including source code characteristics, identifying dependencies, recovering "as-is" legacy system architecture. Techniques to obtain the legacy information range from manual inspection of development history, interviewing developers (if any) and current users to automated reengineering techniques. Techniques to obtain the legacy information range from manual inspection of development history, interviewing developers (if any) and current users to automated reengineering techniques [Canfora et. al., 2007], this process also including for instance, business process understanding, reverse engineering, architectural recovery can be used often with tool support to generate system artifacts.

Although of the challenges that founded during the process of legacy understanding (lack of knowledge and resources, complexity of codes, not updated documents,…), the *LSU* phase does not only assist at creating an inventory of the existing features within the legacy applications, but also facilitates the decomposition of the legacy applications with the aim to maximize reusability. Hence, LSU is essential to the success of legacy to SOA migration [Seacord et. al., 2003], [O'Brien et. al., 2005].

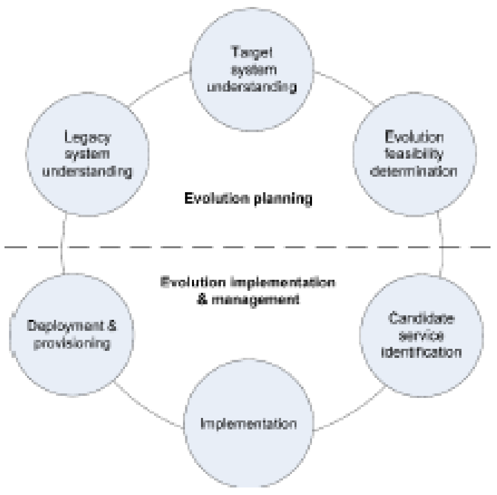

Target system understanding
Legacy system understanding
Evolution feasibility determination
Evolution planning
Evolution implementation & management
Deployment & provisioning
Candidate service identification
Implementation

Figure 8-A: The evaluation framework (Khadka et al., 2013)

The current practices of the research papers in this migration phase represents that, using reverse engineering technique to decomposing the legacy functions and codes is essential to understand the legacy functionalities, however acquiring knowledge and skills from the end users experiences, developers, system and industry experts is still very significant tool to understand the legacy system, this concept is founded in [Nasr et. al., 2011], [Murer et. al., 2011] [Lewis et. al., 2005], [Khadka et. al., 2011], and [Lewis et. al., 2008], and figure 8-B and 8-C depicts these techniques.

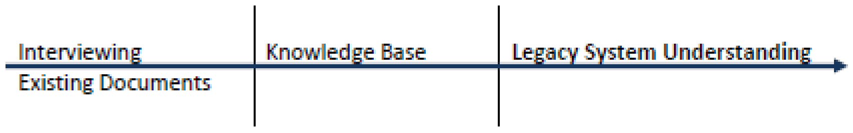


Figure 8-B: Soft Knowledge Technique

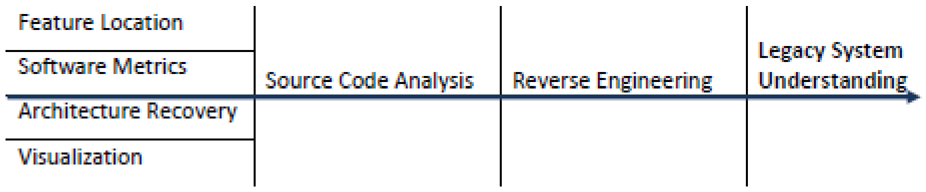


Figure 8-C: Reverse Engineering Technique

In legacy migration, reverse engineering techniques are used to understand the legacy functions via applying deeply source code analysis such as:

- **Architectural Recovery,** that used to extract information/views of a software system from the lower-level artifacts such as source code [Lewis et al., 2008], [Lewis et al., 2005], [O'Brien et al., 2005], [Cuadrado et al., 2008], and [Zhang et al., 2005].

- **Feature location,** used in identifying functional units in a source code, which peace of code represent specific business or technical task [Chen et al., 2005], [Millham, 2010], [Vemuri, 2008].

- **Software Metrics**, have been extensively used; [Sneed, 2008], [Sneed, 2009] measured the size, complexity and quality of legacy programs in terms of modularity, reusability, maintainability metrics, these measurements would support to understand how given legacy system will be ready for modernization process.

- **Source code visualization,** a technique to visualize static and animated forms of software artifacts such as source code and their elements and dependencies, these visualizations would support understand of the systems functionalities [Geet et al., 2007], [Cuadrado et al. ,2008], [Zillmann et al., 2011].

2.8.2 SOA Target System Understanding (SOA-TSU)

SOA target system understanding, this phase aims to understand the SOA key principles, architecture, and environment. Define the main SOA components to be design, and which technology, standards to be used. Also, in this phase some issues like performance, security, governance, and others SOA characteristics to be discussed. [Lewis, et al., 2005] argue that the target architecture largely determine the reusability of the existing legacy components. The other crucial factor that indicates the importance of the target system understanding phase is the fact that legacy applications have undergone numerous bug fixes and over the years they have been efficient, reliable and responsive to the daily business of the enterprise [Bennett, 1995].

SMART method [Lewis et al., 2008], [Lewis et al., 2005] provide guidelines for developing the SOA target architecture based on the legacy components and to assess them with the stakeholder by taking into account various functional and non-functional characteristics of the target system. The SOAMIG method [Zillmann et al., 2011] describes the importance of service design as a part of target system understanding, which is the result of forward engineering (design of the target architecture and the orchestration of services) and reverse engineering (potential features from the previous point of *Legacy System Understanding*). [Cuadrado et al., 2008] explain the selection of specification and service platform to preserve maintainability and interoperability nonfunctional characteristics.

2.8.3 Migration Feasibility Study (MFS)

Understanding the complexity level of the current legacy system, and understanding the architecture design and the new functionalities of the target system would support to shape the feasibility degree of the migration process from different perspectives. The feasibility assessments are carried out at a **technical, economical and organizational** level. The technical assessment includes measuring the code complexity of the legacy system in terms of cohesion, coupling, reusability and abstraction (Reddy, et al., 2009). Economical assessment includes determining economic feasibility of the evolution, for instance by using the cost benefit analysis, as suggested by Sneed (H. M. Sneed, 1995a). This analysis of technical and economic feasibility will provide to the organization most of necessary information to considering whether its business goals will be achieved via SOA migration project or not.

Cost-Benefit Analysis (CBA) proposed by [Sneed, 1995] for determining migration feasibility. The CBA technique is used by [Khadka et al., 2011], [Sneed, 2009], and [Sneed, 2008]. [Umar & Zordan, 2009] extended the CBA model to include the migration costs, which facilitates decision making in choosing a migration strategy. The SMART [Lewis et al., 2005] method uses Options Analysis for Re-engineering (OAR) to determine the so called migration feasibility decision point.

2.8.4 Candidate Service Identification (CSI)

Legacy software is often modified and enhanced by people who did not develop it. Poor documentation and lack of appropriate resources (e.g. developers, architects) make the understanding of source code a hard task. In such a scenario, identifying the potential services and service-rich areas in a legacy code is definitely a challenging task [Zillmann et al., 2011] and [Kontogiannis et al., 2008]. Identifying candidate services is an important activity in the context of legacy to SOA migration as this activity enables reusability and leveraging the existing legacy features [Lewis et al., 2005]. A plethora of methods are reported [Gu & Lago, 2010], [Arsanjani et al., 2008]) to identify potential services.

This phase is focusing on determine which legacy source code' area is potential for re-use' services, various techniques can be used for this purpose. For instance, design pattern recovery, cluster analysis techniques, architectural reconstruction, feature location, concept analysis, and source code visualization can be used to identify the needed/re-used services. CSI is categorized into two approaches:

- ***Top-down,*** started initially by modeling the business process based on the requirements and then the process is subdivided into sub-processes until these can be mapped to legacy functions, this approach is used by [Alahmari et al., 2010], [Fuhr et al., 2011], [Ricca & Marchetto, 2008], and [Zillmann et al., 2011].
- ***Bottom-up*** approach utilizes the legacy code to identify services using various techniques such as information retrieval [Aversano et al., 2008], concept analysis [Zhang et al., 2006], business rule recovery [Ricca & Marchetto, 2008], source code visualization [Geet et al., 2007].

2.8.5 Implementation Process (Imp. Process)

This phase is one of the crucial phases of the process in which the migration is technically realized. This phase provides techniques to extract/leverage the legacy code as services. [Almonaies et al., 2010] classified the implementation strategies into four categories figure 9, migration strategy is mostly selected based on two factors, cost/business value against technical capabilities.

- ***Replacement*** in which a legacy application is replaced entirely with a commercial off-the-shelf (COTS) product.

- ***Integration*** in which the existing legacy application is accessible via an interface, and exposing its functionalities via web services.

- ***Redevelopment*** in which the entire legacy application is re-developed into SOA.

- ***Migration*** in which a legacy application is gradually moved to SOA with reusing the legacy components.

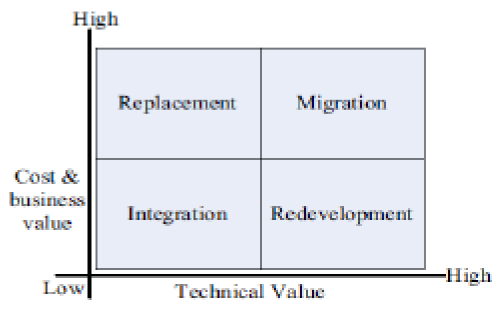


Figure 9: Migration Strategies (Almonaies et al., 2010)

The implementation techniques used in legacy to SOA migration can be broadly grouped into code level and architecture level. Figure 10 [Khadka et al., 2013] depicts various implementation techniques (non-exhaustive) that are used in legacy to SOA migration.

The **code level** group is further divided into various techniques that have been used in legacy to SOA migration such as slicing [Khadka et al., 2011], [Zhang et al., 2006], [Marchetto and Ricca, 2008], [Chen et al., 2009], wrapping [Sneed, 2008], [Sneed, 2009], [Marchetto and Ricca, 2008], refactoring [Cuadrado, 2008], and code transformation [Zillmann et al., 2011]. In general, wrapping is presented as fast, less risky, economical and easy implementation technique. At the **architecture level**, graph transformation techniques are used by [Heckel et al., 2008] and [Fuhr et al., 2011]. Some of the other techniques being used in legacy to SOA migration are inspired by model-driven engineering [Fuhr et al., 2011], [Alahmari et al., 2010].

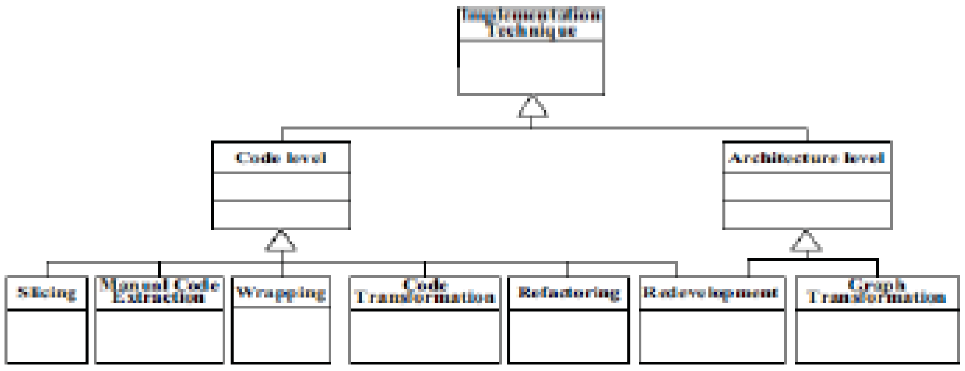


Figure 10: Implementation Techniques (Khadka et al., 2013)

2.8.6 Services Deployment and Provisioning (SD&P)

In this phase the exposed service is deployed in the SOA framework infrastructure, and tested to determine if the expected legacy functionality is exposed correctly as a service. A successful deployment then requires service provisioning that includes activities such as publishing and discovering services in a catalog, maintaining Quality of Services (QoS), versioning, testing, and evolution of services [Khadka et al., 2011b]. Also this phase includes post migration activities that are crucial to the SOA environment. Services are loosely coupled computation entities [Papazoglou et al., 2008] and proper management of these entities throughout their life cycle is an absolute requirement [Papazoglou et al., 2007]. Activities such as service discovery, maintaining QoS of services, testing and evolution of services that lead to the proper functioning of the services ensure that the SOA environment operates reliably and efficiently.

Several research papers are reported on service discovery domain [Rambold et al., 2009] in which the authors present categories of service discovery approaches and compare those approaches. While a survey of service testing approaches has been reported by [Canfora & Di Penta, 2009]. And for overall service evolution, various approaches have been reported for managing the evolution of services, such as [Andrikopoulos et al., 2008] presents a service evolution management framework to identify changes and introduce version control mechanism for services; [Papazoglou, 2008] present a theoretical approach for addressing the service evolution problem; and [Fang et al., 2007] describe a service versioning mechanism to assist service evolution.

Consolidation Method Characteristics: The method is considered that the SOA migration process is required the consolidation of both planning and execution migration aspects. And considered also that the legacy to SOA migration is not only a complex technical endeavor, but it also involves various organizational and business perspectives. The method classified the migration stages into two categories of six stages: Legacy System Understanding, Target system Understanding, Evolution Feasibility Determination, Candidate service Identification, Implementation, and Deployment & provisioning.

The consolidation method is used the Software Metrics, to measure the size, measure the complexity and quality of legacy programs in terms of modularity, reusability, maintainability metrics, these measurements would support to understand how given legacy system will be ready for modernization process.

2.9 ARTIST Project

ARTIST is stands for Advanced Software based-service provisioning and Migration of legacy Software, the project is established to prepare, support and increase the competitiveness of the European Software and Services Industry in a global Cloud and Software as a Service (SaaS) business environment, ARTIST develops a set of methods, tools and techniques that facilitate the transformation and modernization of non-cloud software assets and businesses. The project creates tools to assess, plan, design, implement and validate the automated evolution of non-cloud software to SaaS and the Cloud Computing delivery model

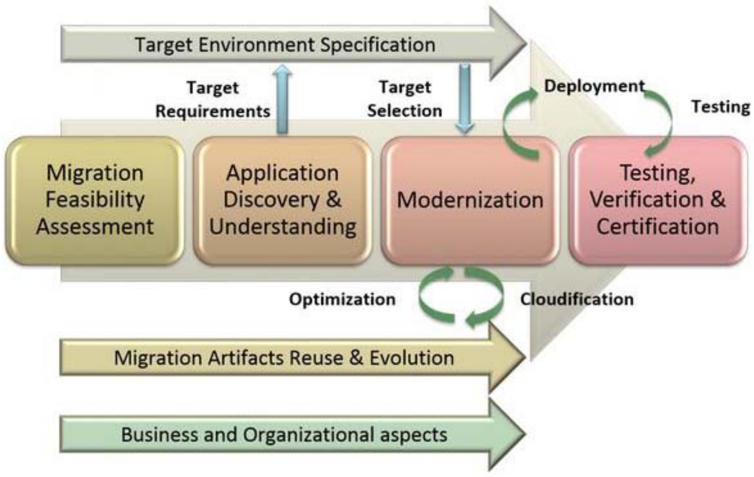

Target Environment Specification
Target Requirements
Target Selection
Deployment
Testing
Migration Feasibility Assessment
Application Discovery & Understanding
Modernization
Testing, Verification & Certification
Optimization
Cloudification
Migration Artifacts Reuse & Evolution
Business and Organizational aspects

Figure 11: ARTIST Project (On Line, 2015)

The ARTIST approach focuses on migration of non-cloud/Legacy software applications to new computing paradigms like service oriented architectures and cloud solutions. The ARTIST project is intend to develop the tools and methods to overcome significant obstacles and reduce the costs and risks associated with the migration of non-cloud software Applications to new IT paradigms like SOA-based technologies and Cloud platforms. The project starts with assess, plan, design, then perform and finally validate and verify the migration of non-cloud software systems into different target framework(s), (Online, http://www.artist-project.eu/objectives, Feb-2015).
The ARTIST Methodology consists of four major phases, figure 11:

- **Pre-migration:** the phase of studying the technical and economic feasibility to perform migration/ modernization of the legacy system.
- **Migration:** executing and implementing migration phase by using reverse engineering and forward engineering techniques in order to deploy the legacy system in the cloud includes the verification (V&V) of the final system.
- **Provisioning:** checked if both technical and business objectives have been achieved to increase customer confidence in the new system.
- **Evolution:** post-implementation phase includes all needed maintenance activities of the application after migration to the cloud.

The artist migration phases will be described in some details as follows:

2.9.1 Pre-migration

The first step in this pre-migration phase is to analyze how mature the application is in terms of technology and business. The analysis of the current situation and the ideal situation supports ARTIST to perform a gap analysis, described in terms of a technical feasibility analysis and the business feasibility analysis. The results obtained in both the feasibility and business analysis will guide decision makers in the strategy of migrating a legacy application or start from scratch.

2.9.2 Migration

This phase concerned in implement the transition activities to create new function or improve the existing legacy functions. ARTIST considered the quality check of the migrated system from the functional and non-functional concerns such as performance or security. The migrated system has to function similarly to the legacy system and needs to perform at least equally to the old system. The non-compliance of any of these requirements may cause project failure.

2.9.3 Provisioning/Post-Migration

ARTIST considered that one of the major problems in such migration project is the reluctance of customers to consume new software offered as a service. ARTISt recommend to demonstrate the provided services to the consumers, which gurantee good quality, secure, load-balanced, trustable, etc. ARTISt adopted the use of the Certification Model that analyses:

- Organization (processes, products, financial aspects, and service continuity),
- Service offered (security, administration, support, QoS, SLA, service operational maturity) and
- Application (functionality, usability, maintenance).

2.9.4 Evolution

ARTIST considered in this phase all maintenance activities needed for the application once the migration to the cloud and the adaptation to SaaS paradigm have been completed. The generation of models with different levels of refinement using MDE techniques (Model Driven Engineering) will facilitate the understanding of the whole system and will facilitate any platform migration process and / or forward engineering that may be necessary.

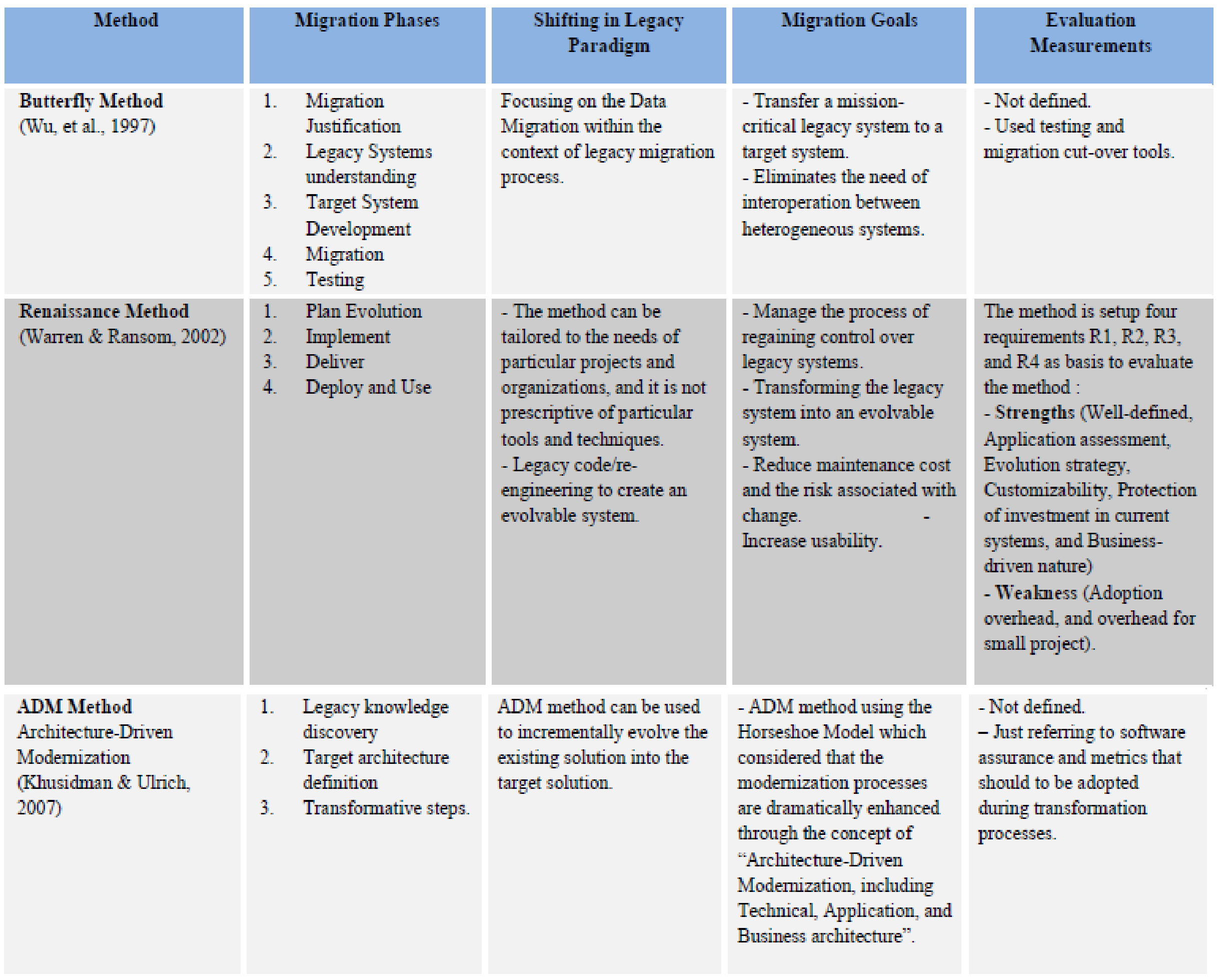
3 CHARACTERISTICS AND GAPS OF EXISTING MIGRATION FRAMEWORKS

In section 2 we have illustrated and discussed the relative work of SOA Migration frameworks and approaches, review selected publications from 1997 to 2015 and see the state of the art of transforming the legacy systems to SOA processes. Several different frameworks are illustrated varying from high level abstraction of migration phases up to re-engineering processes that targeting legacy architecture modernization, including model-driven based approach, reverse/forward engineering methods, SOMA, SMART, SOA-MF, SOAMIG, and others migration architectures. Then the structured framework is displayed to consolidate the proposed migration phases from planning and implementation perspectives.

Also, due to the importance of the migration implementation topic we have illustrated the most migration strategies and techniques used in SOA migration projects including Replacement, Wrapping, Redevelopment, and Migration strategies, and described service identification strategies which is the most important and critical function used in SOA migration, and finally we discovered the common implementation architectures used in services integration and communications such as web services integration, direct Database access, adapters, and Enterprise Service Bus ESB.

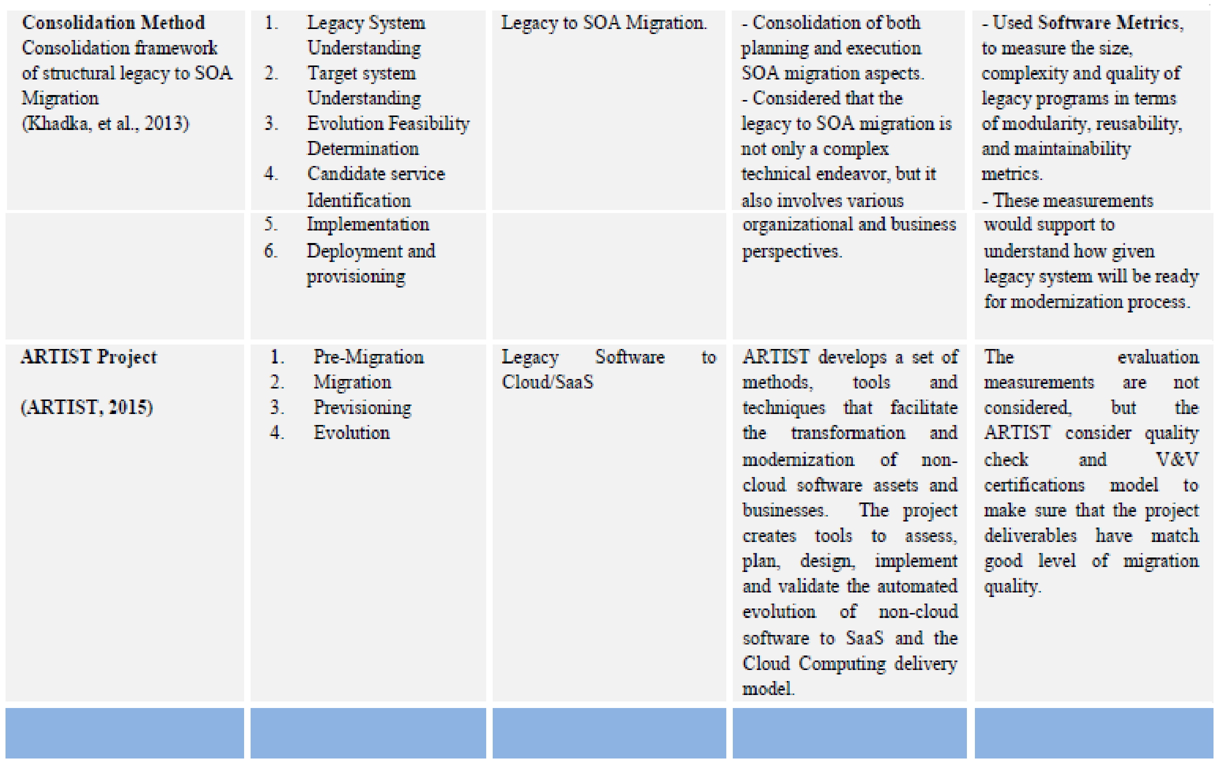
For summarization, we conduct a comparison Table 1 between these presented approaches on four subjects (Migration Phases, Legacy Paradigm Change, Migration Goals, and the Adopted Evaluation Measurements) to evaluate each method's efficiency and quality as follows:

Table 1: SOA Migration Frameworks - Comparison Table



Method	Migration Phases	Shifting in Legacy Paradigm	Migration Goals	Evaluation Measurements
Butterfly Method (Wu, et al., 1997)	1. Migration Justification 2. Legacy Systems understanding 3. Target System Development 4. Migration 5. Testing	Focusing on the Data Migration within the context of legacy migration process.	- Transfer a mission-critical legacy system to a target system. - Eliminates the need of interoperation between heterogeneous systems.	- Not defined. - Used testing and migration cut-over tools.
Renaissance Method (Warren & Ransom, 2002)	1. Plan Evolution 2. Implement 3. Deliver 4. Deploy and Use	- The method can be tailored to the needs of particular projects and organizations, and it is not prescriptive of particular tools and techniques. - Legacy code/re-engineering to create an evolvable system.	- Manage the process of regaining control over legacy systems. - Transforming the legacy system into an evolvable system. - Reduce maintenance cost and the risk associated with change. - Increase usability.	The method is setup four requirements R1, R2, R3, and R4 as basis to evaluate the method : - **Strengths** (Well-defined, Application assessment, Evolution strategy, Customizability, Protection of investment in current systems, and Business-driven nature) - **Weakness** (Adoption overhead, and overhead for small project).
ADM Method Architecture-Driven Modernization (Khusidman & Ulrich, 2007)	1. Legacy knowledge discovery 2. Target architecture definition 3. Transformative steps.	ADM method can be used to incrementally evolve the existing solution into the target solution.	- ADM method using the Horseshoe Model which considered that the modernization processes are dramatically enhanced through the concept of "Architecture-Driven Modernization, including Technical, Application, and Business architecture".	- Not defined. – Just referring to software assurance and metrics that should to be adopted during transformation processes.

SOMA Method (Arsanjani et al. 2008, Fuhr et al. 2011)	1. Business Modeling 2. Solution Management 3. Service Identification 4. Service Specification 5. Service Realization 6. Service Implementation 7. Service Deployment	SOMA method provided to shift the software paradigm from OO object oriented to SOA service oriented architecture.	Increase business agility and integration to implement a service oriented solution.	- SOMA provides support of monitoring and management of business processes and performance monitoring in the production environment. - SOMA also provides linkages to runtime monitoring and management aspects.
SMART Method Service Migration and Reuse Technique (Lewis 2005, SEI 2008)	1. Establish context and feasibility decision 2. Define candidate services 3. Describe existing capabilities 4. Describe target environment 5. Analyze the Gap 6. Develop strategy	SMART is an approach for making decisions on the Migration of legacy components to services.	- Analyzes the viability of reusing legacy components as the basis for services. - SMART consists of four elements: 1- Semantic understanding of legacy system. 2- Gathering information via interviews to assure broad and consistent coverage of the factors that influence the cost, effort, and risk in migration to services. 3- SMART tools to analyze migration data. 4- Artifact Templates for output products of the migration processes.	Considered concrete analysis of the migration feasibility, risk, and cost involved.
SOA-MF Method SOA Migration Framework (Razavian and Lago, 2010)	1. Reverse Engineering 2. Transformation 3. Forward Engineering	Legacy to SOA environment.	SOA-MF addressing the migration of legacy systems to SOA	- Not defined. - The method focused on the notion of tier plays an important role in positioning and classifying the various migration approaches. - SOA-MF Considering four levels of abstraction including code, basic design elements, composite design element, and concept.
SOAMIG Method SOA Migration (Zillmann et al., 2011)	1. Preparation 2. Conceptualization 3. Migration 4. Transition	SOAMIG's characteristics position it in the field of legacy system migration to SOA environment.	- SOAMIG process is to present a generalized, highly iterative, software migration process, heavily based on code transformation. - SOAMIG process distinguishes four organizational phases exposing important milestones in migration projects.	- Not defined. - Used testing and migration cut-over tools.
SOA Migration (Krupi Patel and Leena Ragha 2013)	1. Architecture Recovery 2. Analysis 3. Mapping 4. Transformation	Legacy to Service-Oriented.	Present a new approach to migrate a legacy system to a new SOA platform. It consists of four steps: Architecture Recovery, Analysis, Mapping and Transformation.	- Not Defined.



Consolidation Method Consolidation framework of structural legacy to SOA Migration (Khadka, et al., 2013)	1. Legacy System Understanding 2. Target system Understanding 3. Evolution Feasibility Determination 4. Candidate service Identification 5. Implementation 6. Deployment and provisioning	Legacy to SOA Migration.	- Consolidation of both planning and execution SOA migration aspects. - Considered that the legacy to SOA migration is not only a complex technical endeavor, but it also involves various organizational and business perspectives.	- Used **Software Metrics**, to measure the size, complexity and quality of legacy programs in terms of modularity, reusability, and maintainability metrics. - These measurements would support to understand how given legacy system will be ready for modernization process.
ARTIST Project **(ARTIST, 2015)**	1. Pre-Migration 2. Migration 3. Previsioning 4. Evolution	Legacy Software to Cloud/SaaS	ARTIST develops a set of methods, tools and techniques that facilitate the transformation and modernization of non-cloud software assets and businesses. The project creates tools to assess, plan, design, implement and validate the automated evolution of non-cloud software to SaaS and the Cloud Computing delivery model.	The evaluation measurements are not considered, but the ARTIST consider quality check and V&V certifications model to make sure that the project deliverables have match good level of migration quality.

4 ISSUES FOR A NEW QUALITY-BASED SOA MIGRATION APPROACH

4.1 Motivations

In many cases the Legacy to SOA Migration is recommended as a new software modernization approach to add new business and technology features or to avoid the limitations and problems that might cause by the siloed nature of existing legacy applications which manifest themselves as islands of data, automation, and security [Massoud 2012]. The cost effective approach to overcome this island behavior and its consequences is to keep these systems running and to base a new solution on the existing applications portfolio, and leverage integration as a mechanism for accessing the existing capabilities. Service-Orientated approach with emphasis on reusability and flexibility is often the optimum solution to improve the legacy functionalities, and to suppo rt the initial business integration pilot projects to expand their scope to become enterprise-wide. SOA becomes the preferred approach for delivering business integration platform.
Despite the fact that the SOA migration process is successeded to make the legacy systems running under modern paradigm and derived benefits from its new features, there are some of legacy limitations and problems are still exist, and some of the migration outcomes are not efficient as expected. Therefore, SOA migration process should to be executed under qualified approach that consider the quality characteristics in all its migration phases. This paper is presented to discuss how to design, implement, and evaluate new quality-based SOA-migration framework that mitigate the repeating of the legacy issues in the new SOA environment.

4.2 Intentions and Considerations

As per our literature survey in the field of SOA migration frameworks, and based on the research Gap Analysis mentioned in the previous point, we considered that the most critical quality directions that formulate the quality level in SOA-Migration process can be classified into three topics. The following figure 12 displays the proposed qua lity requirements model in SOA-Migration:

- SOA Architecture Design and Enablement

- SOA Process Integrity

- SOA Quality Evaluation and Measurements

4.2.1 SOA Architecture Design- SAD

The quality requirements in target system planning and design phase are intend to choose the architecture design and its related SOA technologies, which eventually plays an important role in the efficiency and adaptability of the future SOA system. Basically, target system understanding can be viewed from two perspectives: functional characteristics and technical characteristics:

- The functional characteristics include the potential functionalities to-be evolved from the legacy code. This process is referred to service design and application composition. It also defines to what level of granularity the services are to be defined and, accordingly, the orchestration of the services has to be managed to support business processes. Various functional and non-functional properties should also be considered, such as maintainability, interoperability, responsiveness, performance, security, and availability.

- The technical characteristics of the target environment include service technology (SOAP or REST-based), messaging technologies, communication protocols, service description languages, and service discovery mechanisms.

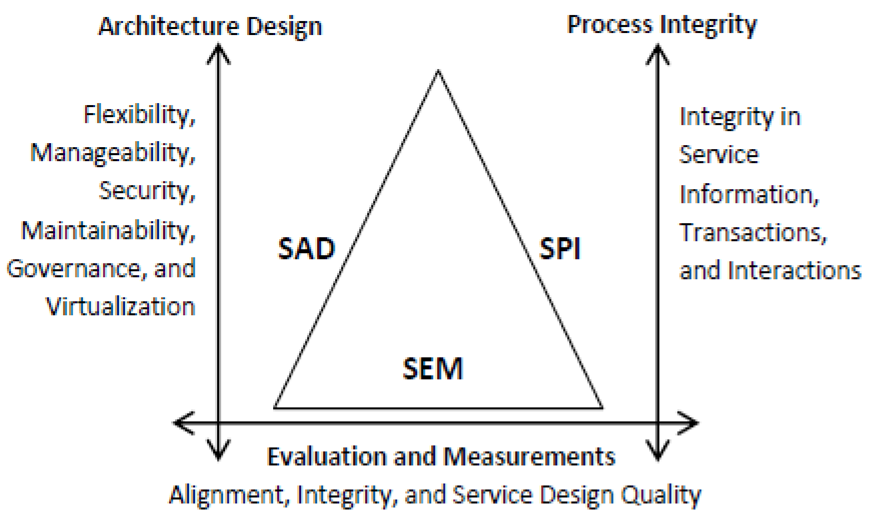

Architecture Design
Process Integrity
Flexibility, Manageability, Security, Maintainability, Governance, and Virtualization
SAD
SPI
SEM
Integrity in Service Information, Transactions, and Interactions
Evaluation and Measurements
Alignment, Integrity, and Service Design Quality

Figure 12: Quality Requreiments Model in SOA Migration

The paper proposed model figure 12 (Quality Requirements Model in SOA Migration) is considered six major characteristics that shape the power of SOA architecture design, including Flexibility, Manageability, Security, Maintainability, Governance, and Virtualization.

4.2.2 SOA Process Integrity - SPI

SOA process integrity is the ability to conduct reliable business activity in a consistent SOA environment with seamless integration at every interacted and participated service. In general, process integrity is the critical component of SOA implementation, the ability to synchronize between services, human tasks, information, applications, domains and users in a secure, scalable SOA environment. Business must be agile enough to deliver the same reliability, consistency and predictability in an open service-oriented system as in a tightly coupled closed system. In SOA, the role of migration/integration is not only to bridge the islands legacy systems, but also to deal with the process integrity/consistency issues. Process integrity has three main elements:

Transaction integrity: Ensures that individual updates of business and IT resources are linked and processed as a single unit of work, all completing successfully or being rolled back in case of technical or business failure.
Interaction integrity: Ensures that elements of people's interactions with business and IT systems are intact and remembered wherever and whenever those interactions occur in secure, scalable, and reliable environment.
Information integrity: Helps deliver trusted, secured information to business processes, regardless of delivery channel, operational platform (IT or people), and information lineage, in which the information to be meaningful, accurate, correctness, and aligned.

So, the quality requirements model recommended to apply some sort of integrity mechanisms to avoid the pitfalls that could be encountered when extending SOA infrastructure from limited-scope projects to a broader enterprise wide implementation, and describes how the considering of the integration quality can help to deliver on the promises of service-orientation approach.

4.2.3 SOA Evaluation Measurements - SEM

After converting legacy systems to be services by transformation the legacy code (migration approach) or by exposing/interfacing the legacy functionalities (integration approach), these services have to be deployed. Some necessary activities are required to manage and control the behavior of services during usage**.** Monitoring the service behavior is very important to maintain the service performance, validation, integrity, etc… Service controlling has been a research challenge in the SOA domain due to the dynamic uses of the services in the SOA context. Build business logic using the legacy services is needed to be controlled to validate the integration process workflow, services input/output, and services data mapping. Another important topic is service quality measurements, measuring the services description, security, data consistency, and others measurements that support the services quality. The mentioned quality model is considered these kinds of research issues by providing several considerations during the design phase, and provides integration evaluation metrics to measure and evaluate the evolved services.

5 ISSUES FOR A NEW SOA MIGRATION FRAMEWORK - SMF

SOA Migration Framework (SMF) is a method that describes the migration processes to transform the legacy applications to work under SOA environment. As a software development life-cycle method for developing SOA-based solutions, or any solution using service-oriented principles, SMF defines key techniques and describes the roles on a SOA migration project includes activities and tasks, the input and output artifact work products for legacy-SOA transition, and the prescriptive methods, guidance and recommendations needed for detailed analysis, design, implementation, deployment, and measurements of services, components, and flows to build a robust and efficient SOA environment.

5.1 SMF Roadmap

SOA Migration Framework (SMF) is designed based on the analysis, considerations and derivations shown in figure 13. The E4 approach, establish, extract, evaluate and execute is an appropriate measurement approach in order to qualify the software development and maintenance involving migration processes [Ebert 2007]. SMF adopted E4 approach during the migration phases to make sure that the migration process is running under qualified methodology.

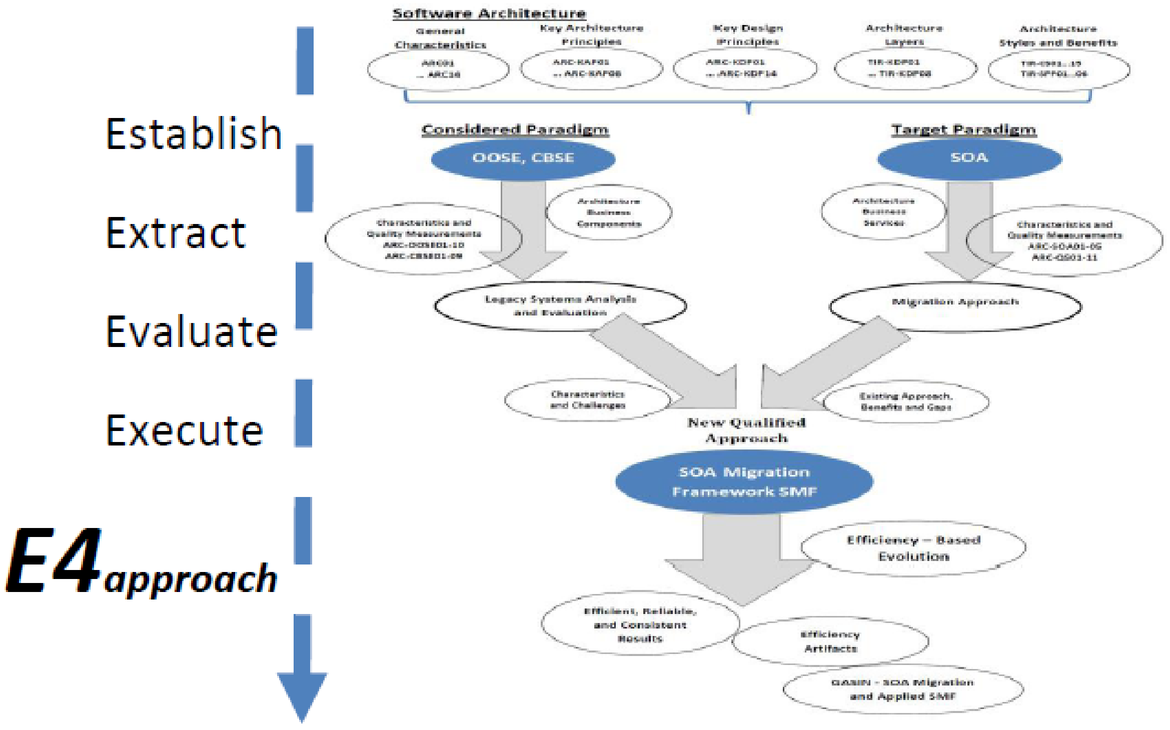

Establish
Extract
Evaluate
Execute
E4approach
Software Architecture
General Characteristics
ARC01 ... ARC16
Key Architecture Principles
ARC-KAP01 ... ARC-KAP08
Key Design Principles
ARC-KDP01 ... ARC-KDP14
Architecture Layers
TIR-KDP01 ... TIR-KDP08
Architecture Styles and Benefits
TIR-CS01..15 TIR-SPF01..06
Considered Paradigm
OOSE, CBSE
Target Paradigm
SOA
Characteristics and Quality Measurements ARC-OOSE01-10 ARC-CBSE01-09
Architecture Business Components
Architecture Business Services
Characteristics and Quality Measurements ARC-SOA01-05 ARC-QS01-11
Legacy Systems Analysis and Evaluation
Migration Approach
Characteristics and Challenges
Existing Approach, Benefits and Gaps
New Qualified Approach
SOA Migration Framework SMF
Efficiency – Based Evolution
Efficient, Reliable, and Consistent Results
Efficiency Artifacts
GASIN - SOA Migration and Applied SMF

Figure 13: The SMF of method derivation

A simple example of the E4 application shows the following description of two kinds of project management improvement in figure 14 (from [Ebert 2007]):

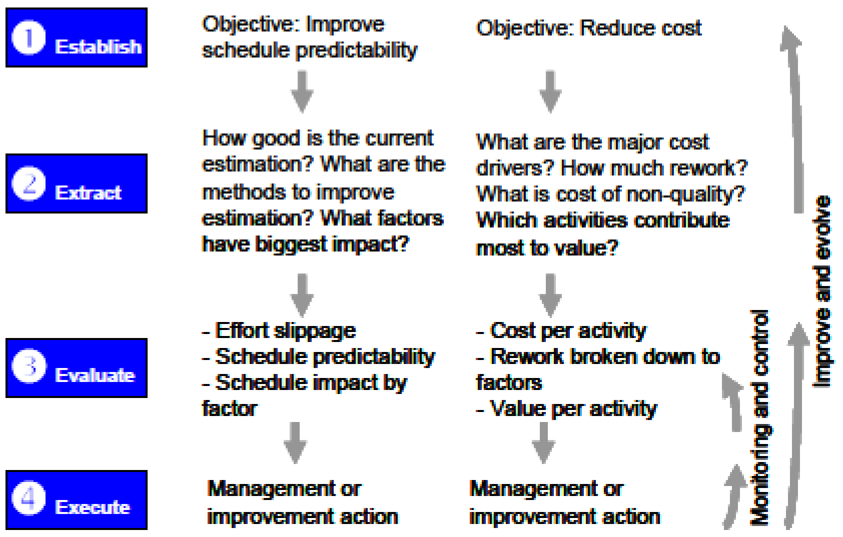

1 Establish
2 Extract
3 Evaluate
4 Execute
Objective: Improve schedule predictability
How good is the current estimation? What are the methods to improve estimation? What factors have biggest impact?
- Effort slippage
- Schedule predictability
- Schedule impact by factor
Management or improvement action
Objective: Reduce cost
What are the major cost drivers? How much rework? What is cost of non-quality? Which activities contribute most to value?
- Cost per activity
- Rework broken down to factors
- Value per activity
Management or improvement action
Monitoring and control
Improve and evolve

Figure 14: Simple Example of E4 measurement process

The application and practical use of SOA Migration Framework (SMF) is based on the following steps and phases characterizes in figure 15

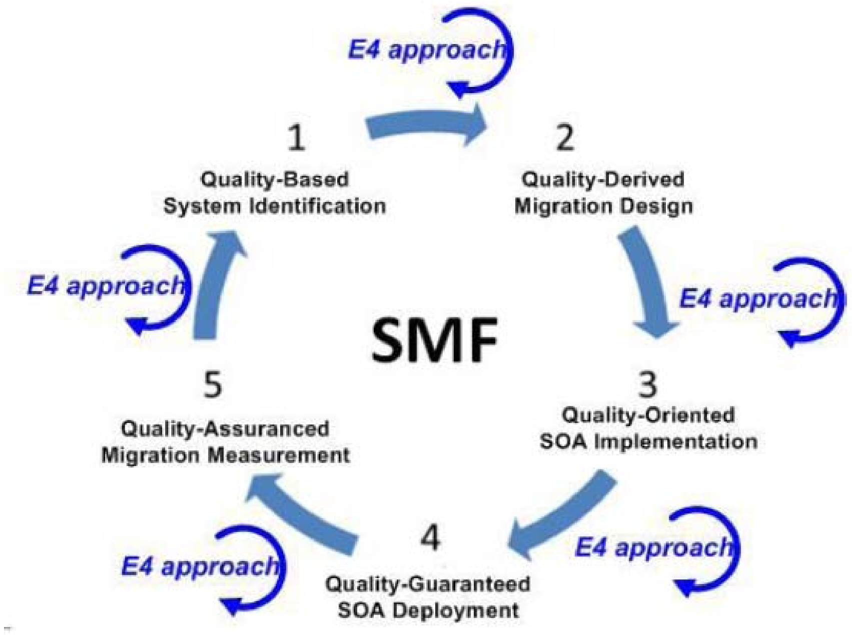

E4 approach
1
Quality-Based System Identification
2
Quality-Derived Migration Design
E4 approach
E4 approach
SMF
5
Quality-Assuranced Migration Measurement
3
Quality-Oriented SOA Implementation
4
Quality-Guaranteed SOA Deployment
E4 approach
E4 approach

Figure 15: The SMF cycle of method application

5.3 SMF Migration phases and Major Activities

SMF consists of five qualified migration phases:

Qualified-Based System Identification

This phase is presented as a migration planning phase, interested in four elements (Feasibility Study, Legacy Code Analysis, Service Identification, and Service Specification) that addresses the issues of making the migration feasibility study, this phase is aim to decide if the existing legacy systems are needed and ready to be migrated to SOA solution from the technical and business perspectives, discuss which technical methodology and approach is a proper one to be used to understand the existing legacy code and its component’s structures and functionalities, and also this phase is concerning in how to identify the candidate part of the legacy code to be re-presented as a reusable service in the target SOA architecture.

Qualified-Derived Migration Design

SOA target system design and understanding phase is aim to understand the SOA key principles, architecture, and environment. Define the main SOA components to be designed, and which technology, standards to be used. Also, in this phase some issues like performance, security, governance, integrity, and others SOA characteristics to be discussed. Design phase support to facilitate the representation of the desired SOA architecture, enables the design of the target architecture with major components of the SOA environment, standards to be used, quality of service (QoS) expectations, and interaction patterns between services.

In **SMF** the design phase is considered that the architecture design should align between the legacy systems characteristics and the enterprise business models toward efficient migration process. So, to achieve this objective, SMF provides the required architecture tools for the design components including SOA Reference Architecture, Enterprise Semantic Context and Information models, Enterprise Business Process Model, Integrity Enablements, and Goverence Controls.

Qualified-Oriented SOA Implementation

Several techniques are presented to implement the migration process. However, SMF adopted the wrapping technique (fastest, less risky and cost effective technique) to migrate the legacy systems by interfacing it to other software via web services. It is a black-box modernization technique, since it focuses on the interface of the legacy systems, hiding the complexity of its logic. Also, the re-engineering technique is target to add the SOA

capabilities and functionalities to the existing legacy systems via reverse engineering, and redesigning the existing software.
SMF is adopted the integration strategy to migrate to SOA architecture, and use the mix between the re-engineering and wrapping strategies to implement the services needed to build the migration solution. Integration enables disparate resources to share business data. **SMF** provides its implementation approach in the following steps:

1- Validate the migration business drivers

2- Determine which architectural layer to perform the integration activities

3- Identify the implementation access type

4- Designing Service Implementation

5- Identify the integration application form

6- Implement the integration architecture

Qualified-Guaranteed SOA Deployment

After implemented the necessary services which exposing the candidate legacy functionalities, the exposed services are then deployed in the service infrastructure and tested to determine if the expected functionalities are formed and integrated correctly. A successful deployment is require a service provisioning that includes activities such as publishing and discovering services in a repository, maintaining Quality of Services (QoS), versioning, testing, and evolution of services that lead to the proper functioning of the services and ensure that the SOA environment operates reliably and efficiently.

SMF considered in the guaranteed the deployment and versioning phase by allowing service implementations to evolve without breaking existing consumers, leading to more services loosely coupled, minimize the impact of versioning, and reduce the amount of deployed code. In SOA, service versioning considered the coexistence of multiple versions of the same service, which allows each consumer to use the target version that it is designed and tested for. In this multiple coexisting versions of the same service, the system allows for the independent life cycles of services and their consumers and minimizes the overall impact of changes to new version.

Qualified-Assurance Migration Measurements

Having deployed services is not enough to move the existing legacy enterprise systems from the islands platforms to SOA environment. **SMF** is considered that in order to complete the migration project efficiently and successfully, there is a need to right kind of services, well-designed and properly built services, efficient services communication, and reliable services that be able to satisfy the current and the future business requirements. Proposal SMF migration framework is focuses on how we can improve the efficiency factors on SOA-Migration.

SMF describes the migration process as follows:

$$_{\text{OOSE/CBSE}}^{\text{System\_1}}ARC \Rightarrow {}_{\text{SOSE}}^{\text{System\_1}}ARC \quad (1)\,^{*}\,,\; {}_{\text{OOSE/CBSE}}^{\text{System\_2}}ARC \Rightarrow {}_{\text{SOSE}}^{\text{System\_2}}ARC\;(2)\,^{*}\text{, etc...}$$

SMF describes the migration metrics and measurement as follows:

$$_{\text{scheduledusage}}^{\text{quality\_improvements}}MP_{\text{research\_approach}}^{\text{CBSE/OOSE}} \xrightarrow{\textit{quality\_criteria}} {}_{\text{scheduledusage}}^{\text{quality\_improvements}}MP_{\text{research\_approach}}^{\text{SOSE}} \quad (3)\,^{**}$$

Also, SMF describes the quality improvements as follows:

- Efficiency Measurements ∈ {cost ∨ *performance* ∨ *flexibility*}

- Consistency Measurements ∈ {Data Validation ∨ *Service Interactions* ∨ *Service Transactions*}

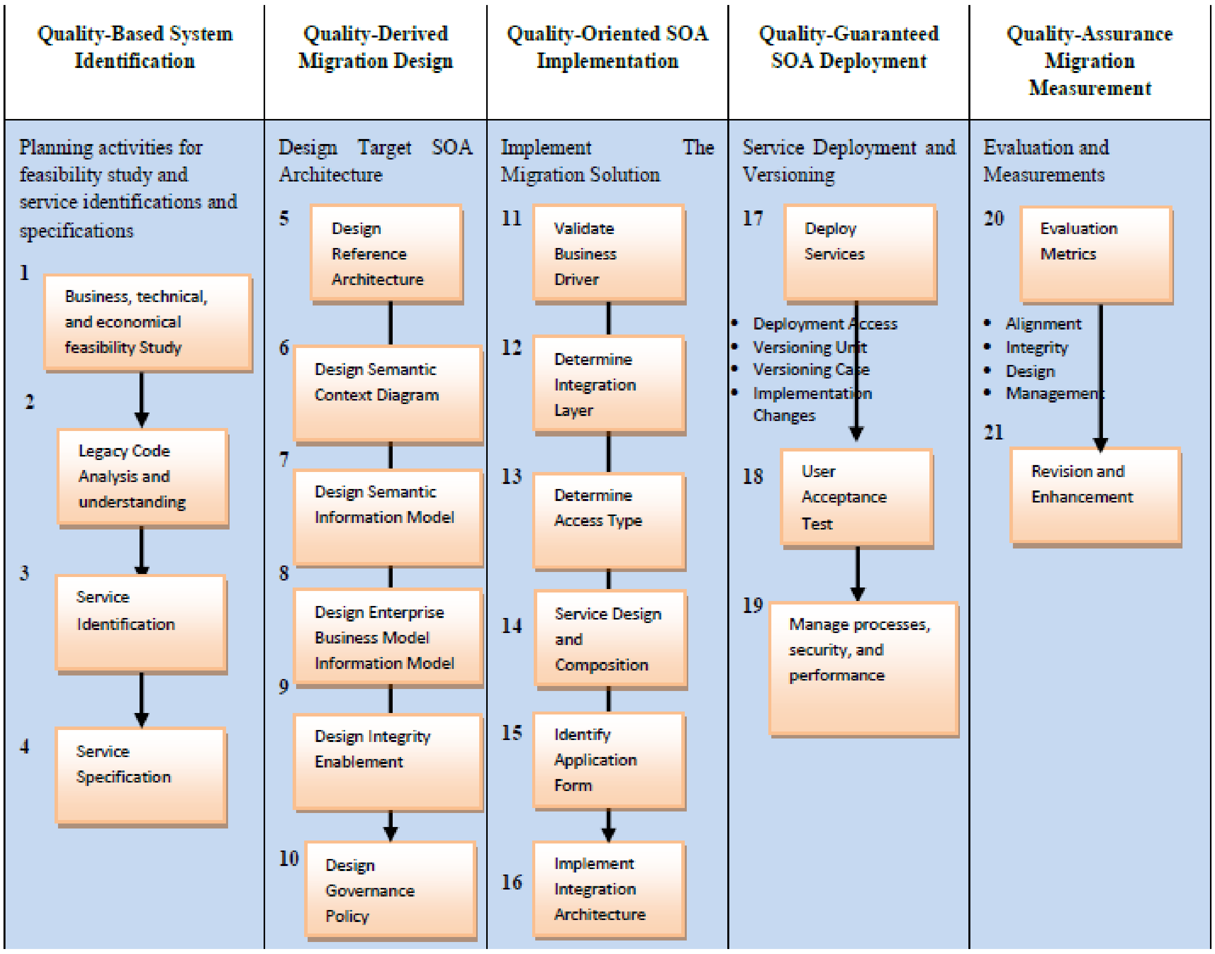
- Level of Service-Interoperability ∈ {Input Validation ∨ Output Validation}

- Level of Loose-Coupling ∈{Independent Services ∨ Dependent Services}

- Characteristics of Island Systems ∈ {Overlapping Object ∨ Limited Function ∨ Semantic dissonance ∨ Inconsistent Data ∨ Insufficient Business Workflow ∨ Lack of Enterprise Data and Business Model}

The detailed activities for every phase of the SMF approach are characterized in the following table 2:

Table 2: SMF Migration phases and Activities

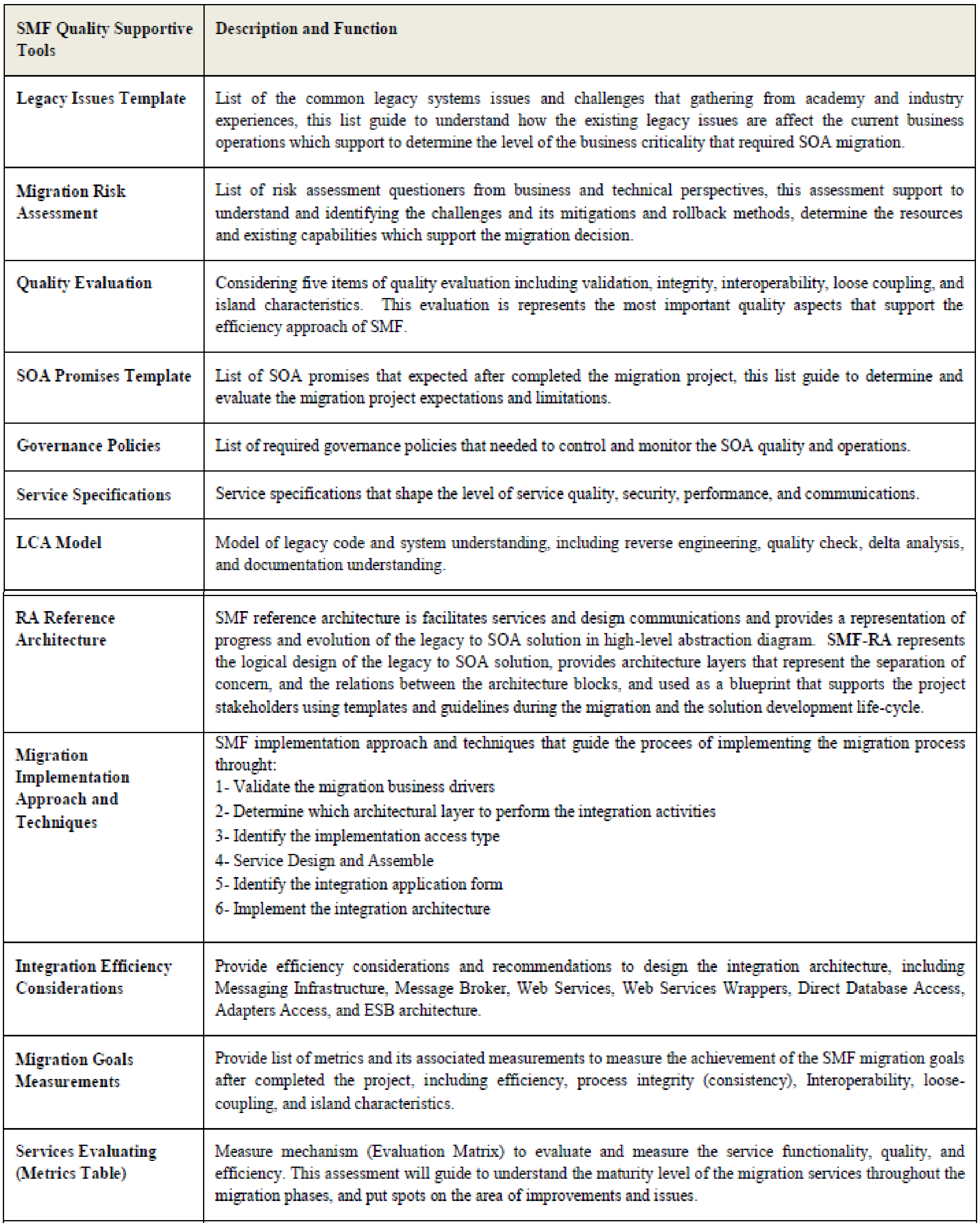


Quality-Based System Identification	Quality-Derived Migration Design	Quality-Oriented SOA Implementation	Quality-Guaranteed SOA Deployment	Quality-Assurance Migration Measurement
Planning activities for feasibility study and service identifications and specifications	Design Target SOA Architecture	Implement The Migration Solution	Service Deployment and Versioning	Evaluation and Measurements
1 Business, technical, and economical feasibility Study	5 Design Reference Architecture	11 Validate Business Driver	17 Deploy Services	20 Evaluation Metrics
2 Legacy Code Analysis and understanding	6 Design Semantic Context Diagram	12 Determine Integration Layer	• Deployment Access • Versioning Unit • Versioning Case • Implementation Changes	• Alignment • Integrity • Design • Management
3 Service Identification	7 Design Semantic Information Model	13 Determine Access Type	18 User Acceptance Test	21 Revision and Enhancement
4 Service Specification	8 Design Enterprise Business Model Information Model	14 Service Design and Composition	19 Manage processes, security, and performance	
	9 Design Integrity Enablement	15 Identify Application Form		
	10 Design Governance Policy	16 Implement Integration Architecture		

**ARC* (Software Architectur)
***MP* (Measurement Process)

SMF added further aspects that support the migration quality SMF Quality supportive tools and methods using several artifacts products,methods, recommendations, guidelines, and new design model. Table 3 display in brief some of these suppo rtive items:

Table 3: The principles of SMF suppo rtive tools



SMF Quality Supportive Tools	Description and Function
Legacy Issues Template	List of the common legacy systems issues and challenges that gathering from academy and industry experiences, this list guide to understand how the existing legacy issues are affect the current business operations which support to determine the level of the business criticality that required SOA migration.
Migration Risk Assessment	List of risk assessment questioners from business and technical perspectives, this assessment support to understand and identifying the challenges and its mitigations and rollback methods, determine the resources and existing capabilities which support the migration decision.
Quality Evaluation	Considering five items of quality evaluation including validation, integrity, interoperability, loose coupling, and island characteristics. This evaluation is represents the most important quality aspects that support the efficiency approach of SMF.
SOA Promises Template	List of SOA promises that expected after completed the migration project, this list guide to determine and evaluate the migration project expectations and limitations.
Governance Policies	List of required governance policies that needed to control and monitor the SOA quality and operations.
Service Specifications	Service specifications that shape the level of service quality, security, performance, and communications.
LCA Model	Model of legacy code and system understanding, including reverse engineering, quality check, delta analysis, and documentation understanding.
RA Reference Architecture	SMF reference architecture is facilitates services and design communications and provides a representation of progress and evolution of the legacy to SOA solution in high-level abstraction diagram. **SMF-RA** represents the logical design of the legacy to SOA solution, provides architecture layers that represent the separation of concern, and the relations between the architecture blocks, and used as a blueprint that supports the project stakeholders using templates and guidelines during the migration and the solution development life-cycle.
Migration Implementation Approach and Techniques	SMF implementation approach and techniques that guide the procees of implementing the migration process throught: 1- Validate the migration business drivers 2- Determine which architectural layer to perform the integration activities 3- Identify the implementation access type 4- Service Design and Assemble 5- Identify the integration application form 6- Implement the integration architecture
Integration Efficiency Considerations	Provide efficiency considerations and recommendations to design the integration architecture, including Messaging Infrastructure, Message Broker, Web Services, Web Services Wrappers, Direct Database Access, Adapters Access, and ESB architecture.
Migration Goals Measurements	Provide list of metrics and its associated measurements to measure the achievement of the SMF migration goals after completed the project, including efficiency, process integrity (consistency), Interoperability, loose-coupling, and island characteristics.
Services Evaluating (Metrics Table)	Measure mechanism (Evaluation Matrix) to evaluate and measure the service functionality, quality, and efficiency. This assessment will guide to understand the maturity level of the migration services throughout the migration phases, and put spots on the area of improvements and issues.

References

[Almonaies 2010] Almonaies, A.; Cordy, J.; Dean, T.: *Legacy system evolution towards Service-Oriented Architecture.* SOAME’10, IEEE, pp. 53–62., 2010

[Alahmari 2010] Alahmari, S.; Zaluska, E.; De Roure, D.: *A service identification framework for legacy system migration into SOA, SCC’10*. IEEE, pp. 614–617., 2010

[Andrikopoulos 2008] Andrikopoulos, V.; Benbernou, S.; Papazoglou, M.P.:*Managing the evolution of service specifications*., *AISE*. Springer, pp. 359–374, 2008

[Arsanjani 2008] Arsanjani, A.; Ghosh, S.; Allam, A.; Abdollah, T.; Ganapathy, S.; Holley, K.: *SOMA: A method for developing service-oriented solutions*. IBM Sys. J., vol. 47, no. 3, pp. 377–396, 2008

[ARTIST 2015] ARTIST Project: *Advanced Software-based Service Provisioning Migration of legacy* Software, ARTIST Newsletter, http://www.artist-project.eu, 2015

[Beier 2009] Beier, A.; Uhlig, D.: *Flow Graph Manipulator (FGM): Reverse Engineering Tool.* f¨ur komplexe Softwaresysteme, Softwaretechnik-Trends, 2(29):39–40, 2009

[Bennett 1999] Bennett, K.; Ramage, M.; Munro, M.: *Decision model for legacy systems*. IEE Proc, 146(3):153-159 Software 1999

[Bergey 1999] Bergey, J.; Smith, D.; Weiderman, N.; Woods, S.: *Options Analysis for Reengineering (OAR). Issues and Conceptual Approach* (No CMU/SEI-99-TN-014): SEI, 1999

[Canfora 2007] Canfora, G.; Di Penta, M.: *New frontiers of reverse engineering.* FSE. IEEE, pp. 326–341, 2007

[Canfora 2009] Canfora, G.; Di Penta, M.: *Service-oriented architectures testing: A survey. Software Engineering*, Springer, pp. 78–105, 2009

[Chen 2005] Chen, F.; Li, S.; Yang, H.; Wang, C.-H.; Cheng-Chung Chu, W.: Feature analysis for service-oriented reengineering. *APSEC’05*, IEEE, pp. 8–pp, 2005

[Chen 2009] Chen, F.; Zhang, Z.; Li, J.; Kang, J.; Yang, H.: *Service identification via ontology mapping.* COMPSAC’09, IEEE, pp. 486–491, 2009

[Correia 2007] Correia, R.; Matos, C.; Heckel, R.; El-Ramly, M.: *Architecture migration driven by code categorization. Software Architecture*, 115-122, 2007

[Cuadrado 2008] Cuadrado, F.; Garc´ıa, B.; Dueas, J.; Parada, H.: *A case study on software evolution towards service-oriented architecture.* AINAW’08, IEEE, pp. 1399–1404, 2008

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

[Ebert, 2010] Ebert, J.; Bildhauer, D.: *Reverse Engineering Using Graph Queries, Graph Transformations and Model Driven Engineering*. LNCS 5765, 2010

[Fang 2007] Fang, R.; Lam, L.; Fong, L.; Frank, D.; Vignola, C.; Chen, Y.; Du, N.: *A version-aware approach for web service directory*. *ICWS’07*, IEEE, pp. 406–413, 2007

[Fleurey 2007] Fleurey, F.; Breton, E.; Baudry, B.; Nicolas, A.; Jézéquel, J. M.: *Model-driven engineering for software migration in a large industrial context. Model Driven Engineering Languages and Systems.* 482-497, 2007

[Fuhr 2010] Fuhr, A.; Horn T.; Winter, A.: *Model-Driven Software Migration*. SE 2010, LNI 159:69–80, 2010

[Fuhr 2011] Fuhr, A.; Horn, T.; Riediger, V.; Winter, A.: *Model-driven software migration into service-oriented architectures.* CSRD, vol. 28, no. 1, pp.65–84, 2011

[Geet 2007] Geet, J.; Demeyer, S.: *Lightweight visualisations of COBOL code for supporting migration to SOA. Soft Evol'07*, 2007

[Gimnich 2007] Gimnich, R. SOA Migration: Approaches and Experience. Softwaretechnik-Trends, *27*(1), 13-14, 2007

[Gu 2010] Gu, Q.; Lago, P.: *Service identification methods: a systematic literature review. Towards a Service-Based Internet*, Springer, pp. 37–50, 2010

[Kazman 1998] Kazman, R.; Woods, S.G.; Carri`ere, S.J.: *Requirements for integrating software architecture and reengineering models,* CORUM II, 154, 1998

[Kazman 2002] Kazman, R.; O'Brien, L.; Verhoef, C.: *Architecture Reconstruction Guidelines, 2nd Edition.* (CMU/SEI-2002-TR-034, ADA 421612), Software Engineering Institute, Carnegie Mellon University, 2002

[Khadka 2011a] Khadka, R. : *Service Identification Strategies in Legacy-to-SOA migration.* Paper presented at the Doctoral consortium of the 26th International Conference on Software Maintenance (ICSM'11), 2011

[Khadka 2011b] Khadka, R.; Saeidi, A.; Jansen, S.; Hage, J.; Helms, R.: *An Evaluation of Service Frameworks for the Manangement of Service Ecosystems.* Paper presented at the 15th Pacific Asia Conference on Information System (PACIS'11), Brisbane, Australia, 2011

[Khadka 2012] Khadka, R.; Saeidi, A.; Jansen, S.; Hage, J.: *Legacy to SOA evolution- a systematic literature review in Migrating Legacy Applications: Challenges in Service Oriented Architecture and Cloud Computing Environments,* Ionita, A. D.; Litoiu, M.; Lewis, G. Eds. IGI Global, pp. 40–71., 2012

[Khadka 2013] Khadka, R.; Saeidi, A.; Jansen, S.; Hage, J.: *A Structured Legacy to SOA Migration Process and its Evaluation in Practice. Maintenance and Evolution of Service-Oriented and Cloud-Based Systems* (MESOCA), IEEE 7th International Symposium, pp. 2-11, 23-Sep.2013

[Khusidman 2007] Khusidman, V.; Ulrich, W.: *Architecture-Driven Modernization. Transforming the enterprise Draft* V.5: OMG, 2007

[Lewis 2005] Lewis, G.; Morris, E.; O'Brien, L..; Smith, D.; Wrage, L.: *SMART: The service-oriented migration and reuse technique.* CMU/SEI, Tech. Rep. CMU/SEI-2005-TN-029, Sept 2005

[Lewis 2006] Lewis, G., Morris, E., Smith, D.: *Analyzing the Reuse Potential of Migrating Legacy Components to a Service-Oriented Architecture.* Proceedings of the 10th European Conference on Software Maintenance and Reengineering, (CSMR 2006), Bari, Italy, March, 2224, IEEE Computer Society, 2006

[Lewis 2008] Lewis, G.; Smith, D.: *Service-oriented architecture and its implications for software maintenance and evolution.* FoSM'08, IEEE, pp. 1–10, 2008

[Lewis 2009] Lewis, G.; Smith, D.; Chapin, N.; Kontogiannis, K.: MESOA’09,”SEI, Tech. Rep. 1424448972, 2009

[Marchetto 2008] Marchetto, A.; Ricca, F.: *Transforming a java application in an equivalent web-services based application: toward a tool supported stepwise approach.* WSE’08, IEEE, pp. 27–36., 2008

[Massoud 2012] *Massoud*, A.; Dumke, R.: *Efficient Reference Architecture for Integrated Legacy Applications based SOA.* In: Abran et al.: IWSM/Mensura Proceedings, Assisi, Italy, CPS Publishing Service of IEEE, Session 1B, 2012

[Massoud 2014] *Massoud*, A.: *Process Integrity in SOA Migration.* In: Büren et al.: Praxis der Software-Messung, Shaker-Verlag, Aachen, S. 205-222, 2014

[Matos, 2008] Matos, C.: *Service Extraction from Legacy Systems.* Proceedings of the 4th international conference on Graph Transformations, pp. 505-507, Springer-Verlag, 2008

[Millham 2010] Millham, R.: *Migration of a legacy procedural system to service oriented computing using feature analysis. CISIS’10,* IEEE, pp. 538–543, 2010

[Murer 2011] Murer, S.; Bonati, B.; Furrer, F. J.: *Managed Evolution.* Springer, 2011

[Nasr 2011] Nasr, K. A.; Gross, H.-G.; Deursen, A. van: *Realizing service migration in industry: lessons learned. JSME,* 2011

[O’Brien 2002] O'Brien, L.; Stoermer, C.; Verhoef, C.: *Software Architecture Reconstruction: Practice Needs and Current Approaches* (CMU/SEI-2002-TR-024, ADA407795). Software Engineering Institute, Carnegie Mellon University, 2002

[O’Brien 2005] O’Brien, L.; Smith, D.; Lewis, G.: *Supporting migration to services using software architecture reconstruction. STeP’05* IEEE, pp. 81–91, 2005

[Papazoglou 2007] Papazoglou, M. P.; Traverso, P.; Dustdar, S.; Leymann, F.: *Service oriented computing: State of the art and research challenges.* Computer, vol. 40, no. 11, pp. 38–45, 2007

[Papazoglou 2008] Papazoglou, M.; Traverso, P.; Dustdar, S.; Leymann, F.: *Service oriented computing: a research roadmap. IJCIS*, vol. 17, no 2, pp. 223–255, 2008

[Rambold 2009] Rambold, M.; Kasinger, H.; Lautenbacher, F.; Bauer, B.: *Towards autonomic service discovery: a survey and comparison. SCC’09*, IEEE, pp. 192–201, 2009

[Razavian 2010] Razavian, M.; Lago, P.: *A frame of reference for SOA migration.* Towards a Service-Based Internet, Springer, pp. 150–162, 2010

[Razavian 2011] Razavian, M.; Lago, P.: *A survey of SOA migration in industry*. Service-Oriented Computing, Springer, pp. 618–626, 2011

[Razavian 2012] Razavian, M.; Lago, P.: *A lean and mean strategy for migration to services.* WICSA/ECSA’12. ACM, pp. 61–68, 2012

[Reddy 2009] Reddy, V. K.; Dubey, A.; Lakshmanan, S.; Sukumaran, S.; Sisodia, R.: *Evaluating legacy assets in the context of migration to SOA.* Software Quality Journal, 17(1), 51-63, 2009

[Seacord 2003] Seacord, R. C.; Plakosh, D.; Lewis, G. A.: *Modernizing legacy systems: software technologies, engineering processes, and business practices.* Addison-Wesley Professional, 2003

[Smith 2007] Smith, D.: *Migration of legacy assets to service-oriented architecture environments.* Software Engineering-Companion, 29th International Conference on (pp. 174-175). IEEE., 2007

[Sneed 2009] Sneed, H.: *A pilot project for migrating COBOL code to web services*. *STTT*, vol. 11, no 6, pp. 441–451, 2009

[Teppe 2009] Teppe, W.: *The ARNO Project: Challenges and Experiences in a Large-Scale Industrial Software Migration Project*. 13th CSMR, IEEE CSP: 149–158, 2009

[Truyen 2006] Truyen, F.: *The Fast Guide to Model Driven Architecture. The Basics of Model Driven Architecture (MDA),* Cephas Consulting Corp., 2006

[Umar 2009] Umar, A.; Zordan, A.: *Reengineering for service oriented architectures: A strategic decision model for integration versus migration.* JSS, vol. 82, no. 3, pp. 448–462, 2009

[Warren 2002] Warren, I.; Ransom, J.: *Renaissance A Method to Support Software System Evolution*. Paper presented at the 26th Annual International Computer Software and Applications Conference, 2002

[Wu 1997] Wu, B.; Lawless, D.; Bisbal, J.; Grimson, J.; Wade, V.; O'Sullivan, D., et al.: *Legacy systems migration - a method and its tool-kit framework.* Paper presented at the Joint 1997 Software Engineering Conference and International Computer Science Conference Asia Pacific, 1997

[Zhang 2005] Zhang, Z.; Liu, R.; Yang, H.: *Service identification and packaging in service oriented reengineering.* SEKE’05, pp. 219–26., 2005

[Zhang 2006] Zhang, Z.; Yang, H.; Chu, W.: *Extracting reusable object-oriented legacy code segments with combined formal concept analysis and slicing techniques for service integration*. *QSIC’06* IEEE, pp. 385–392, 2006

[Zillmann 2010] Zillmann, C.; Gringel, P.: *Iterative Zielarchitektur definition in SOAMIG.* SWT-Trends, 2(30):39–40, 2010

[Zillmann 2011] Zillmann, C. ; Winter, A.; Herget, A.; Teppe, W.; Theurer, M.; Fuhr, A.; Horn, T.; Riediger, V.; Erdmenger, U.; Kaiser et al., U.: *The SOAMIG Process Model in Industrial Applications.* CMSR’11, IEEE, pp. 339–342., 2011

[Zimmermann 2010] Zimmermann, Y.; Uhlig, D.; Kaiser, U.: *Tool- und Schnittstellenarchitektur f ̈ur eine SOA-Migration*. SWTTrends, 2(30):66–67, 2010

Established Software Metrics adapting to COSMIC Measurement

Reiner Dumke, Anja Fiegler, Heike Hegewald, Robert Neumann, Cornelius Wille

University of Magdeburg, SML@b

Abstract – *This paper is an extension of our IWSM/Mensura presentation 2014 and discusses the extensions of the COSMIC Function Point method using empirical aspects in order to support the broader application of this method for effort estimation and other software system und processes characterization. The method extensions are based on our experience in different COSMIC applications for embedded systems, agile development, SOA implementations, cloud computing and apps implementation in the last ten years.*

After a short introduction about this well-known COSMIC method, empirical aspects of software products and processes are described and applications of effort estimation based on sizing the quality, technology and methodology are discussed.

1 BASICS OF THE COSMIC FP METHOD AND THEIR APPLICATION

The COSMIC Function Point method (COSMIC FP or CFP method) is a functional size measurement with following characteristics ([Abran 2010], [COSMIC 2014], [Dumke 2010]):

- the CFP method is conform to the international standard for functional size measurement (FSM) as ISO/IEC 14143,
- against the other point metrics, the CFP method can be applied for business software, embedded/real time systems and other modern software system paradigms,
- this method defines a ratio scale functional size with an clear described measurement unit as CFP,
- the basis of CFP method is a I/O counting of the software system functionality,
- the deriving of the CFP value is independent of the software artefacts (as requirements, software models, architectures, programming and testing artefacts, documentations and maintenance artefacts),
- the CFP method itself is an international standard as ISO 19761.

Comparing the software functionality with (mathematical) functions (as e. g. $y = f(x_1, x_2, . . . , x_n)$ or $G_{x1+x2+...+xn}(z) = G_{x1}(z)G_{x2}(z). . . G_{xn}(z)$ etc.), the following extensions for software functionality are essential ([Bundschuh 2008], [Ebert 2007], [Leiss 2007]):

- the software functionality consists (like mathematical functions) of inputs and outputs and the functional operations in order to produce the (user) outputs,
- the software functionality is usual connected with a graphical user interface and different application techniques based on user event models,
- the software functionality is based on different paradigms and technologies for functionality/ algorithm implementation and application like a programming technique *T* including programming language(s) *L* and their grammars *G*, software processors *P* (as compilers, editors, emulators, generators etc.), the programming paradigm *M* (as OOSE, CBSE, SOSE or AOSE), the programming environment or infrastructure *U* (as client/server, Web services, clouds etc.) and the programming experiences *E* (as laws, rules (of thumb), experiments etc.) as $T = (\{L(G)\},\{P\}, M, U, E)$.

The general functional measurement approach using for the CFP method can be characterized as following.

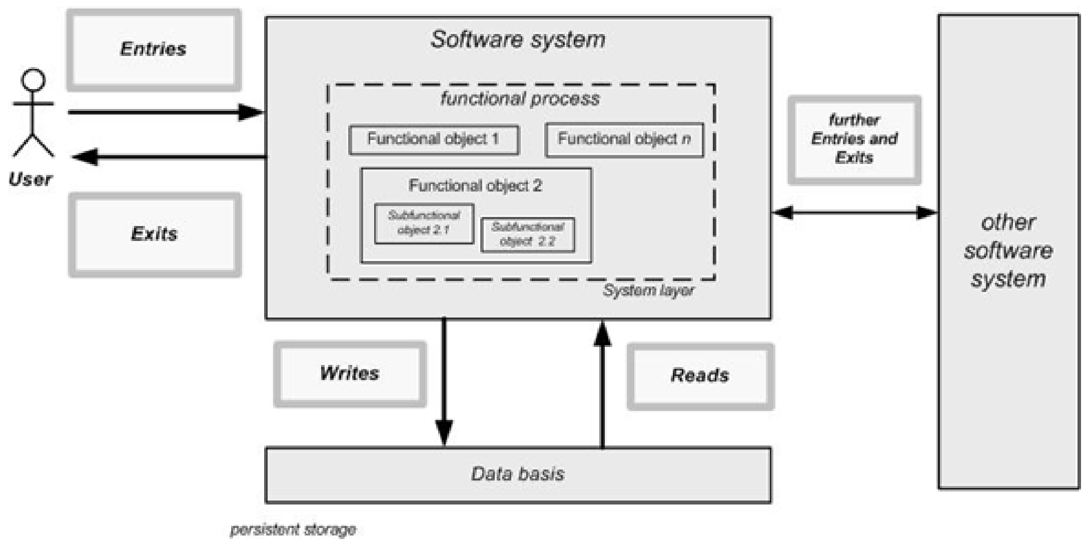


Figure 1**:** COSMIC FP measurement principles

This figure shows the mainly consideration of functionality of the CFP method as an input/ouput or I/O counting. The benefit of this approach is the *"*pure*" functional size measurement*. The general components of the COSMIC FP method are shown in the following figure.

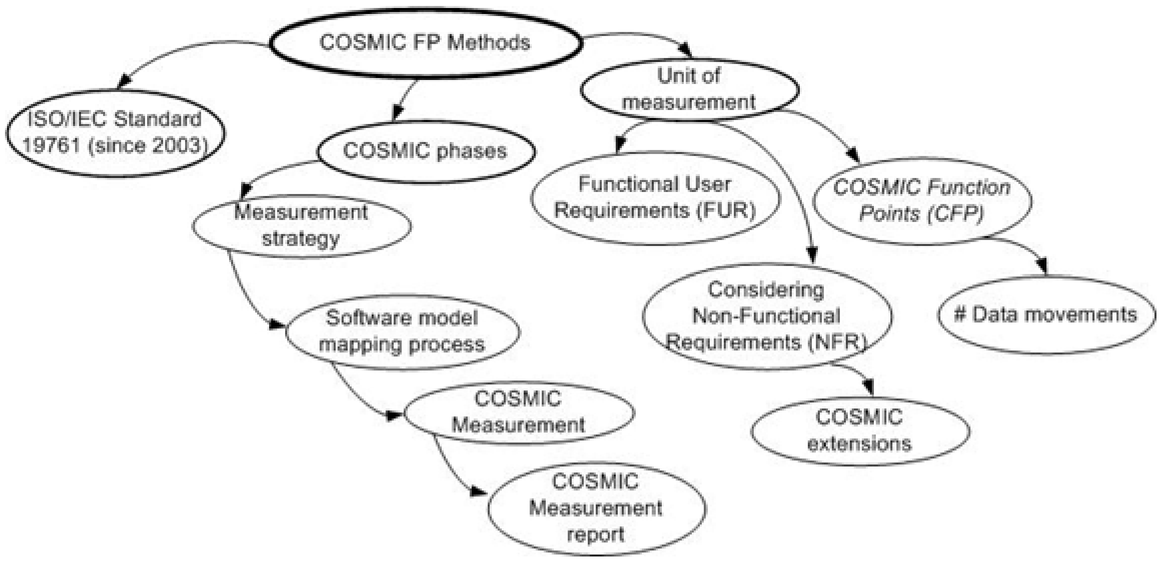

COSMIC FP Methods
ISO/IEC Standard 19761 (since 2003)
COSMIC phases
Unit of measurement
Measurement strategy
Functional User Requirements (FUR)
COSMIC Function Points (CFP)
Software model mapping process
Considering Non-Functional Requirements (NFR)
# Data movements
COSMIC Measurement
COSMIC extensions
COSMIC Measurement report

Figure 2: COSMIC FP method components

The measurement process itself is based on so-called I/O metrics involving the analysis of the data movements in the given functional processes. The elements of I/O counting is given in the following figure.

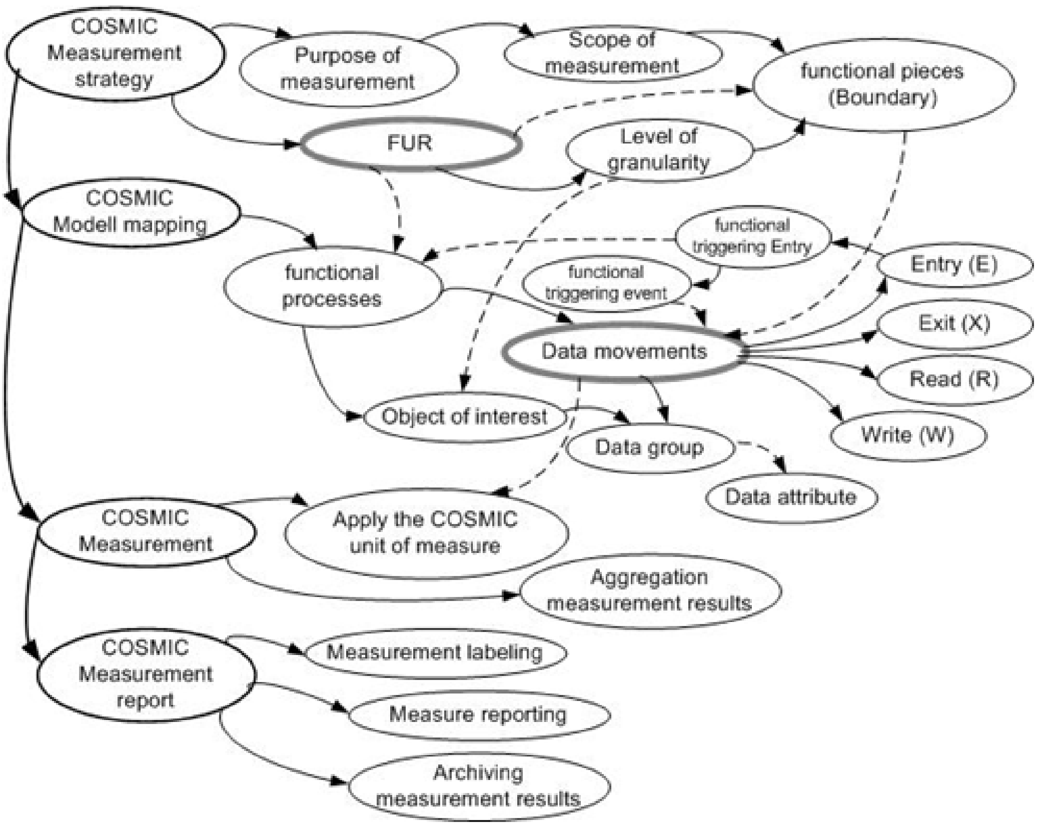


Figure 3: COSMIC FP measurement elements

The typical I/O counting of the CFP method are summarized in the COSMIC patterns by Symons [Symons 2013] (using the usual symbols as *E* for an entry, *X* for exit, *R* for read and *W* for write) ([Schmietendorf 2012], [Schmietendorf 2007], [Schmietendorf 2013], [Schmietendorf 2010], [Wille 2011]).

- the functional size measurement of *embedded or real time systems* are based on sets of *E* and *X* with the possible extensions by *R* and *W* as

$$CFP = |\{E_i, X_j\}| + \#(R, W) \quad (1)$$

- the functional size measurement of *business application* involves any rows of *E* and *X* and sets of *R* and *W* as

$$CFP = \#(\{E_{i1}, X_{j1}\}, ..., \{E_{in}, X_{jn}\}) + |\{R_i, W_j\}| \quad (2)$$

  This characterization includes the typical situations for service oriented systems, apps and cloud computing.

- the measurement of other software systems like *knowledge-based systems or communication systems* can be characterized as

$$CFP = \#(E, X) + \#(\{R_{i1}, W_{j1}\}, ..., \{R_{in}, W_{jn}\}) \quad (3)$$

  or

$$CFP = \#(\{E_{i1}, X_{j1}\}, ..., \{E_{in}, X_{jn}\}^{i}) \quad (4)$$

  and any other more.

The use of the COSMIC generic software model includes any aspects of their arcgitectural design noted in the following figure.

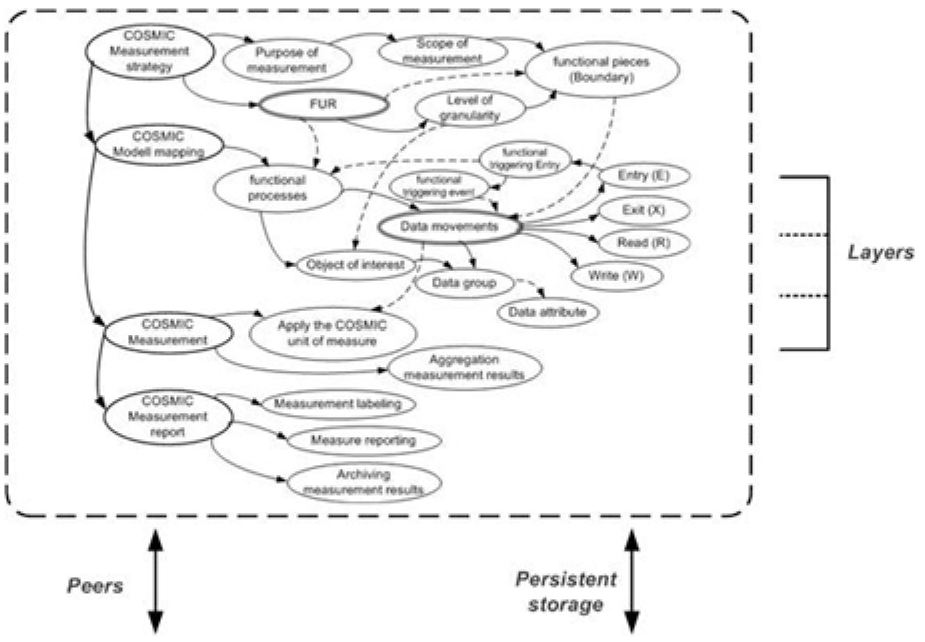


Figure 4: Architectural aspects in COSMIC FP measurement

This kind of COSMIC measurement involves special characteristics for the different classes of software systems and infrastructures characterized in the next figure.

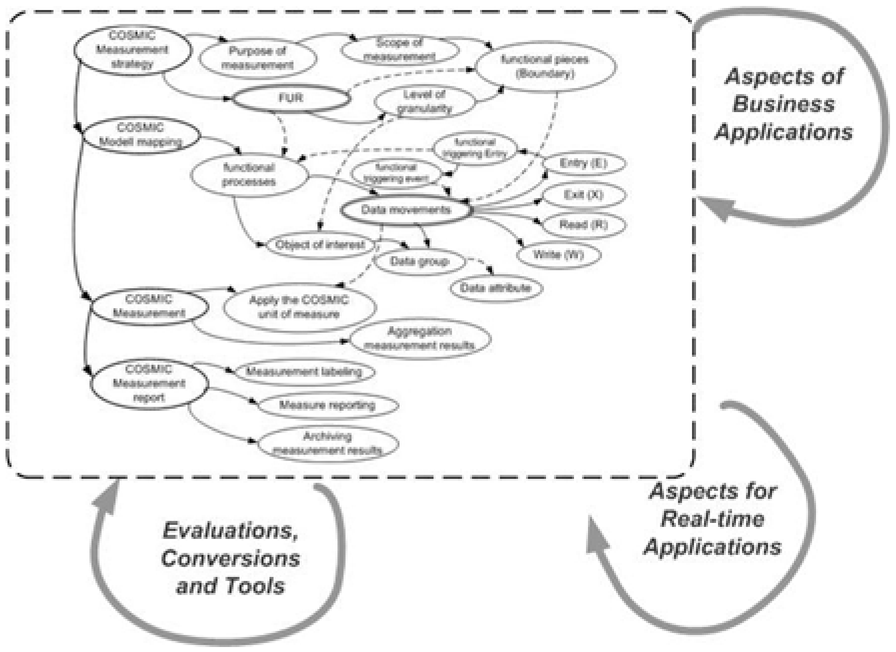

Aspects of Business Applications
Evaluations, Conversions and Tools
Aspects for Real-time Applications

Figure 5: COSMIC measurement for different software systems

Currently, the most applications of functional size measurement methods are the effort/cost estimations. The CFP approach needs any empirical extensions in order to perform any COSMIC effort estimation. It is necessary to involve any (calibration) factors in order to achieve the system related effort characterization. Typical examples of this effort estimation are ([Abran 2010], [Bundschuh 2008], [Dumke 2010], [ISBSG 2012], [Kunz 2007])

- the unit-based characterization of software development effort as (PM as personal month)

$$1\ CFP_{new\_development} \approx 0.07\ PM \qquad (5)$$

- the ISBSG application for effort characterization as

$$1\ CFP_{maintenance} \approx 0.013\ PM \qquad (6)$$

- the deriving of cost estimation by conversion of different FMS measurement like IFPUG FP to COSMIC as

$$1\ FP_{IFPUG} \approx 1.13\ CFP \qquad (7)$$

- the characterization of project duration *D in month* as
  - mainframe computer: $D = 0{,}458 * effort^{0{,}366}$
  - mid-range computer: $D = 0{,}548 * effort^{0{,}360}$
  - PCs: $D = 1{,}936 * effort^{0{,}201}$ (8)

Note, that the details of this characterization are not relevant in the intention of this paper.

2 EMPIRICAL ASPECTS OF SOFTWARE PRODUCTS AND PROCESSES

Empirical aspects are necessary in order to describe and understand software with all their characteristics of software products and software processes ([Chemuturi 2009], [Jones 2007], [Kunz 2007], [Laird 2006]). General empirical aspects are classified as *software size, software quality* and *software complexity.*

The empirical aspect for software sizing leads in their possible problems in successful process management of software system development, maintenance and application. They are some different kinds of software sizes like

$$SIZE = SIZE_{product} \oplus SIZE_{process} \qquad (9)$$

with

$$SIZE_{product} = SIZE_{artefact} \oplus SIZE_{empirical}, \qquad (10)$$

$$SIZE_{artefact} = \{\#requirements, \#models, \#components, \#testCases, \#LOC, \#docPages \text{ etc.}\}, \qquad (11)$$

$$SIZE_{empirical} = \{functional\ size, quality\text{-}based\ size, paradigm\text{-}based\ size, platform\text{-}based\ size\} \qquad (12)$$

and

$$SIZE_{process} = \{\#phases, \#activities, \#resources, \#budgets, \#versions, \#methods \text{ etc.}\}\ . \qquad (13)$$

Considering the software requirements, we can differ between the functional (user) requirements (as *FUR*) and non functional requirements (as *NFR*) where the *NFR* can be classified in quality user requirements (as *QUR*), system/platform user requirements (as *PUR*) and process/project organizational requirements (as *POR*). The summarizing of software requirements is

$$REQ = FUR \cup NFR = FUR \cup \{QUR, PUR, POR\} \qquad (14)$$

The *NFR* requirements can be written in more details (but not completely) as ([7], [10], [19])

$$QUR_{ISO\ 9126} = QUR_{product} \cup QUR_{application}, \qquad (15)$$

with

$QUR_{product}$ = {*FUR, reliability, usability, efficiency, maintainability, portability*}, (16)

$QUR_{application}$ = {*effectiveness, productivity, safety, satisfaction*}, (17)

and

PUR = {*paradigm, architecture, programming technology, software processors, infrastructure*}, (18)

POR = {*development method, life cycle, management aspects, personal resources, CASE tools, COTS, hardware resources*} (19)

and

management aspects = {*timeline, effort, costs, size*}. (20)

Note, that the different kinds of software complexity must be considered in the same manner. Typical kinds of complexity are ([Ebert 2007], [Jones 2007], [Leiss 2007]):

$$COMPL = COMPL_{artefact} \oplus COMPL_{empirical}, \quad (21)$$

with

$COMPL_{artefact}$ = {*problem complexity, model complexity, architecture complexity, program complexity, infrastructure complexity*} (22)

and

$COMPL_{empirical}$ = {*topological complexity, information complexity, diagnostic complexity, data complexity, flow complexity, code complexity, mnemonic complexity, cyclomatic complexity* etc.} (23)

A typical description of these empirical aspect is given in the COCOMO II method (without explanations here) as [Boehm 2000]

QUR_{COCOMO} = {CPLX, DATA, DOCU, RCPX, RUSE etc.}

PUR_{COCOMO} = {PVOL, STOR, TIME, TURN} (24)

POR_{COCOMO} = {ACAP, APEX, LTEX, PCAP, FCIL etc.}

These sets should only demonstrate the (scaling) factors using to execute the project effort based on empirical aspects.

Considering these empirical aspects, the COSMIC-based effort estimation can be characterized in general as

$$effort_{CFP\text{-}based} = \alpha_{POR}^{QUR,PUR} \times size_{CFP-based}^{FUR} \quad (25)$$

with $\alpha_{POR}^{QUR,PUR}$ as scaling factor achieving an approximated estimation using the equations (5) to (8). But, that is not the highly quality of COSMIC size measurement achieving *white-box estimation* where the NFR characteristics are given explicitly and not summarized in one number (as α).

3. EMPRICAL-BASED EXTENSIONS OF THE COSMIC FP METHOD

The main intension is: How we can use the granularity of the COSMIC measurement for empirical evaluation of software systems like effort estimation (as effort measurement) or other system characterization? They are many variants of method extensions. We will discuss any of them in this paper. The following figure characterizes the general extensions of the COSMIC FP measurements.

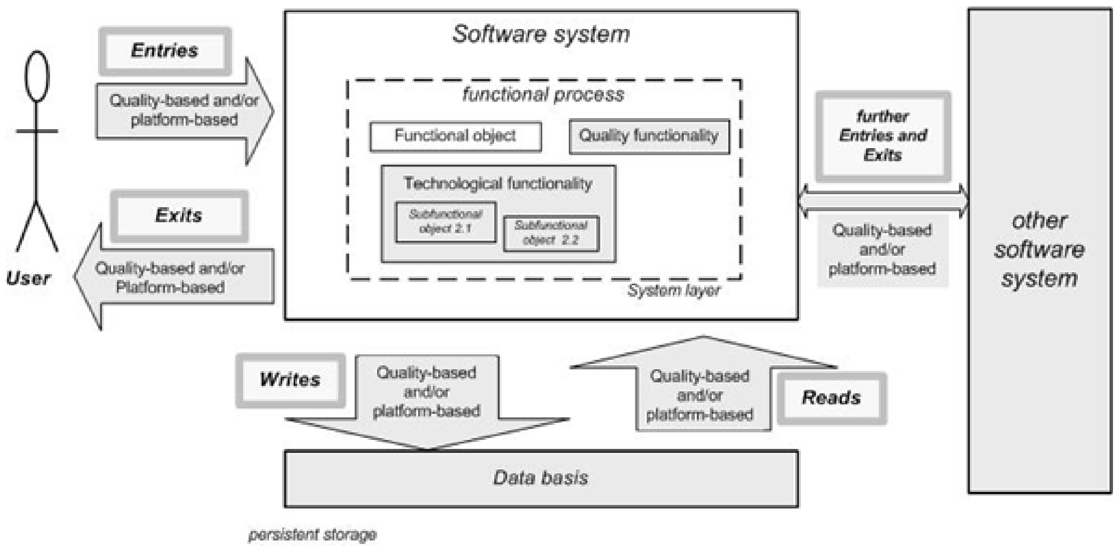

Entries
Quality-based and/or platform-based
Exits
Quality-based and/or Platform-based
User
Software system
functional process
Functional object
Quality functionality
Technological functionality
Subfunctional object 2.1
Subfunctional object 2.2
System layer
further Entries and Exits
Quality-based and/or platform-based
other software system
Writes
Quality-based and/or platform-based
Quality-based and/or platform-based
Reads
Data basis
persistent storage

Figure 6: Extended COSMIC FP measurements

From the empirical aspects point of view, the COSMIC FP method is a functional size measurement where the different counting based on (1) to (4) is

$$CFP = SIZE_{functional} = \#(E, X, R, W) \qquad (26)$$

where $E = E^{FUR}$, $X = X^{FUR}$, $R = R^{FUR}$ and $W = W^{FUR}$. This characterization leads to the question of the measurement of $SIZE_{product}$ as a total/whole software system size which would be necessary in order to estimate/execute the effort and costs.

On the other hand, software process size would be another essential precondition for effort estimation of software development. These questions lead to the consideration of *NFR* that makes the sizing more completely involving E^{NFR}, X^{NFR}, R^{NFR} and W^{NFR}. The principal idea of the empirical extensions should involve more measurements indeed of justifications with any empirical factors. That means the following transformations as

$$effort_{CFP\text{-}based} = \alpha_{POR}^{QUR,PUR} \times size_{CFP-based}^{FUR} \qquad (27)$$

into

$$effort_{CFP\text{-}based} = \alpha_{POR}^{QUR} \times [size_{CFP-based}^{PUR} + size_{CFP-based}^{FUR}] \qquad (28)$$

into

$$effort_{CFP\text{-}based} = \alpha_{POR} \times [size_{CFP-based}^{QUR} + size_{CFP-based}^{PUR} + size_{CFP-based}^{FUR}] \qquad (29)$$

and, finally, into

$$effort_{CFP\text{-}based} = \alpha \times [size_{CFP-based}^{POR} + size_{CFP-based}^{QUR} + size_{CFP-based}^{PUR} + size_{CFP-based}^{FUR}] \qquad (30)$$

Currently, they are a lot of measurements, metrics and/or evaluations in order to count the different sizes. The following table includes anyone of these in a general characterization (see [Abran 2010], [Dumke 2014], [Jones 2010], [Laird 2006], [Munson 2003], [Rud 2006], [Sneed 2010], [Zuse 1998] and (16), (18) and (19)).

Tab. 1: Existing NFR-based size measurements

Empirical aspect	***Metrics/measurements***	***Weaknesses***
Size^{PUR} measurements:		
	Paradigm related metrics for OOSE: #responseForAClass #childrenClasses CBSE: #developedComponents, #involvedCOTS SOSE: max(#serviceOrchestration), #servicesInHierarchyLevel	No ratio scaled for classes; no difference between potential und used functiona-lities Dependencies of technology and component involvements No clear indentification of involved functionalities
	Infrastructure and platform sizes as: #networkNodes, #serviceClusters	System dependencies and unclear functional distributions
Size^{QUR} measurements:		
	Usability measurement: sizeOfHelpComponent #menuButtons	No expression about complete-ness and appropriateness
	Document measurements: #commentsInProgram, sizeOfSystemDoc	Natural and programming lan-guage depended
	Security measurement:: sizeOfPasswordChecking, #firewalls	No ratio scaled in their size of functional extension
	Testability: #testCases #testPathes	No functional relationships and functional coverages
Size^{POR} measurements:		
	Process measurements: #milestones, sizeOfPEERTdiagram	Development method depended and project related
	Ressources measurements: sizeOfCOTS #CASEToolVersions	No ratio scaled and depen-dencies of platforms
	Personal measurements: #teamMembers #functionalityExperts	Dependencies of qualification and effectiveness

These weaknesses could be avoid using a well-defined technology independent and ratio scaled determination of software sizes reasoning in PUR, QUR and POR characteristics. We will consider these extensions in principle as following.

In order to build a COSMIC based measurement, we suggest any mapping of the following COSMIC elements achieving the results for $size_{CFP-based}^{QUR}$. The following figure characterizes these intentions and their involved COSMIC elements.

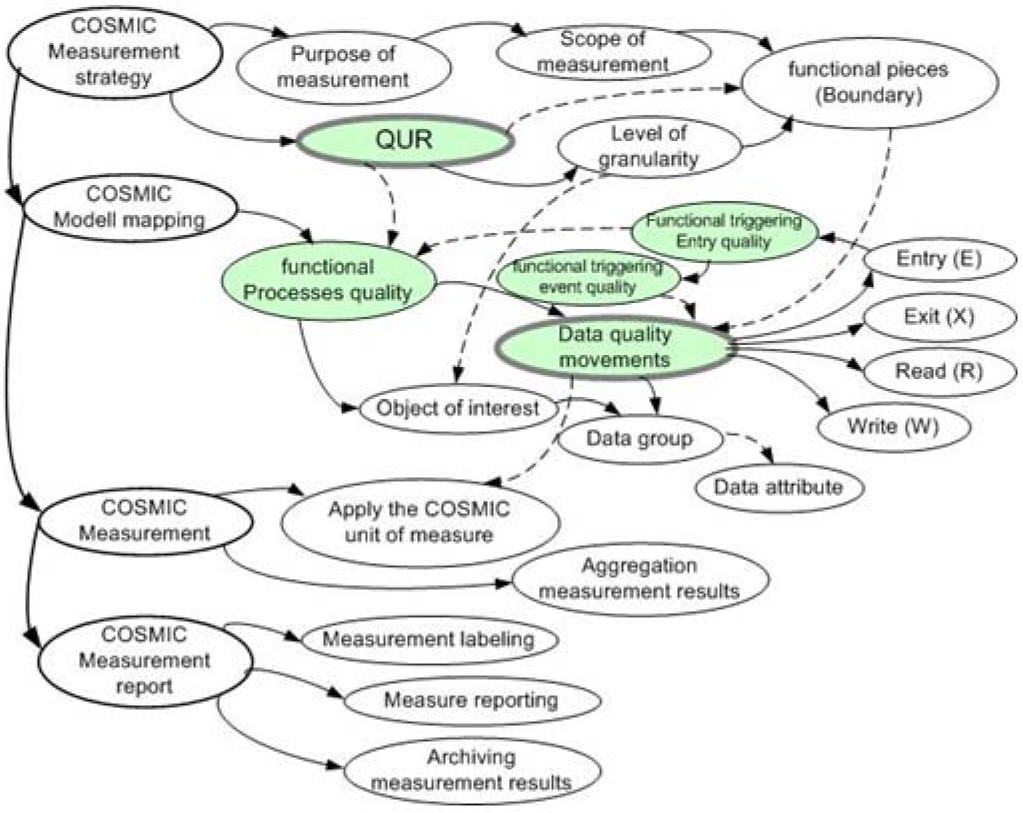
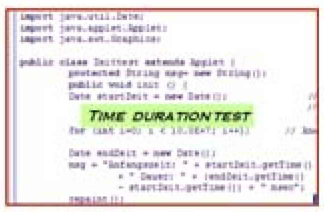

COSMIC Measurement strategy
Purpose of measurement
Scope of measurement
functional pieces (Boundary)
QUR
Level of granularity
COSMIC Modell mapping
Functional triggering Entry quality
functional Processes quality
functional triggering event quality
Entry (E)
Exit (X)
Data quality movements
Read (R)
Object of interest
Write (W)
Data group
Data attribute
COSMIC Measurement
Apply the COSMIC unit of measure
Aggregation measurement results
COSMIC Measurement report
Measurement labeling
Measure reporting
Archiving measurement results

Figure 7: QUR-extended COSMIC FP measurements

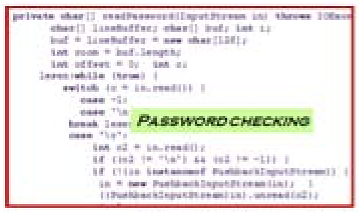
Examples of the quality-based functionalities are characterized in the following figures of Java examples.



```
import java.util.Date;
import java.applet.Applet;
import java.awt.Graphics;

public class Zeittest extends Applet {
        protected String msg= new String();
        public void init () {
        Date startZeit = new Date();
        for (int i=0; i < 10.0E+7; i++);          // [illegible]

        Date endZeit = new Date();
        msg = "Anfangszeit: " + startZeit.getTime()
                + " Dauer: " + (endZeit.getTime()
                - startZeit.getTime()) + " msec";
        repaint();
```



```
private char[] readPassword(InputStream in) throws IOExce
        char[] lineBuffer; char[] buf; int i;
        buf = lineBuffer = new char[128];
        int room = buf.length;
        int offset = 0;  int c;
      loop:while (true) {
          switch (c = in.read()) {
            case -1:
            case '\n
          break loop
          case '\r':
             int c2 = in.read();
             if ((c2 != '\n') && (c2 != -1)) {
             if (!(in instanceof PushbackInputStream)) {
              in = new PushbackInputStream(in);   }
              ((PushbackInputStream)in).unread(c2);
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class Dialogbeispiel extends JFrame implements ActionLis
    Dialogbeispiel () { // Konstruktor fuer das
                        // Dialoghauptfenster
      super("Dialogbeispiel");
      JButton b;
      add(b =
      b.addAct
      add(b = new JButton("Modus 2"), BorderLayout.EAST)
      b.addActionListener(this);
      setSize(400,300);
      setLocation(200,200);
      setVisible(true);
    }
```

QUR performance (as a time duration test in Java)

QUR security (as a password checking in Java)

QUR usability (as a GUI interaction extension in Java)

Figure 8: QUR Java examples

In order to build a COSMIC based measurement for *size* $^{PUR}_{CFP-based}$, we suggest any mapping of the following COSMIC elements achieving the appropriate results. The following figure characterizes the elements that should be modified.

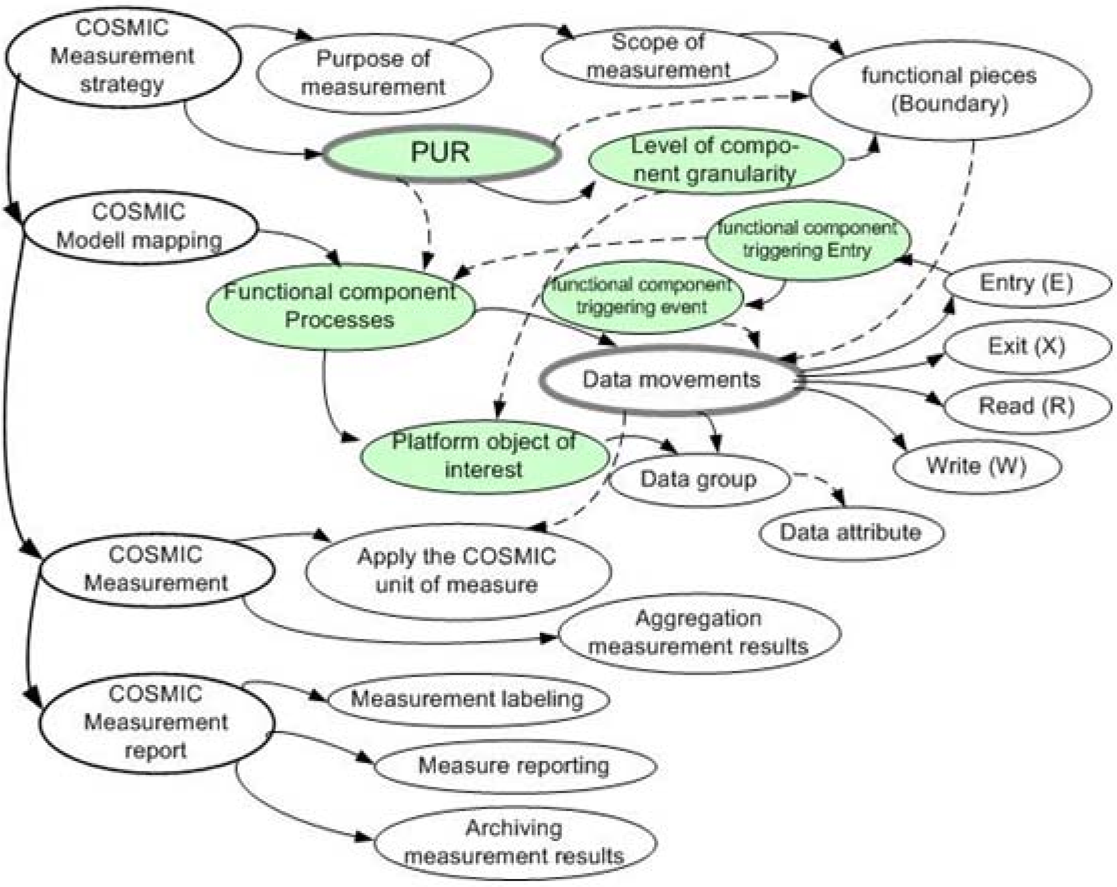
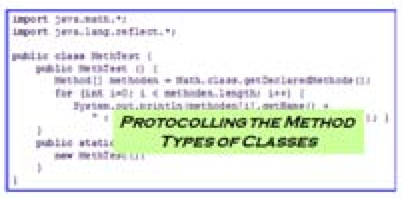

COSMIC Measurement strategy
Purpose of measurement
Scope of measurement
functional pieces (Boundary)
PUR
Level of component granularity
COSMIC Modell mapping
functional component triggering Entry
Functional component Processes
functional component triggering event
Entry (E)
Exit (X)
Data movements
Read (R)
Platform object of interest
Write (W)
Data group
Data attribute
COSMIC Measurement
Apply the COSMIC unit of measure
Aggregation measurement results
COSMIC Measurement report
Measurement labeling
Measure reporting
Archiving measurement results

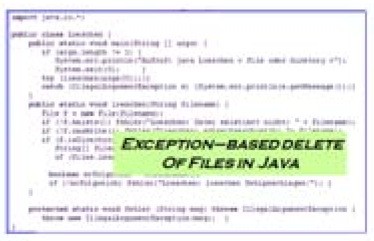
Figure 9: PUR-extended COSMIC FP measurements

Examples of the platform-based functionalities are characterized in the following figures as Java examples.



```
import java.math.*;
import java.lang.reflect.*;

public class MethTest {
    public MethTest () {
        Method[] methoden = Math.class.getDeclaredMethods();
        for (int i=0; i < methoden.length; i++) {
            System.out.println(methoden[i].getName() +
                " : [illegible]
    }
    public static [illegible]
        new MethTest();
    }
}
```



```
import java.io.*;

public class Loeschen {
    public static void main(String [] args) {
        [illegible]
    }
    public static void loeschen(String filename) {
        File f = new File(filename);
        [illegible]
    }

    protected static void fehler (String msg) throws IllegalArgumentException {
        throw new IllegalArgumentException(msg);  }
}
```

PUR class aspects (as a method type protocol in Java)

PUR file management (as exception based file deletion in Java)

Figure 10: PUR Java examples

In order to build a COSMIC based measurement for *size* $^{POR}_{CFP-based}$, we suggest any mapping of the following COSMIC elements achieving the results. The following figure characterizes the involved COSMIC elements in this case.

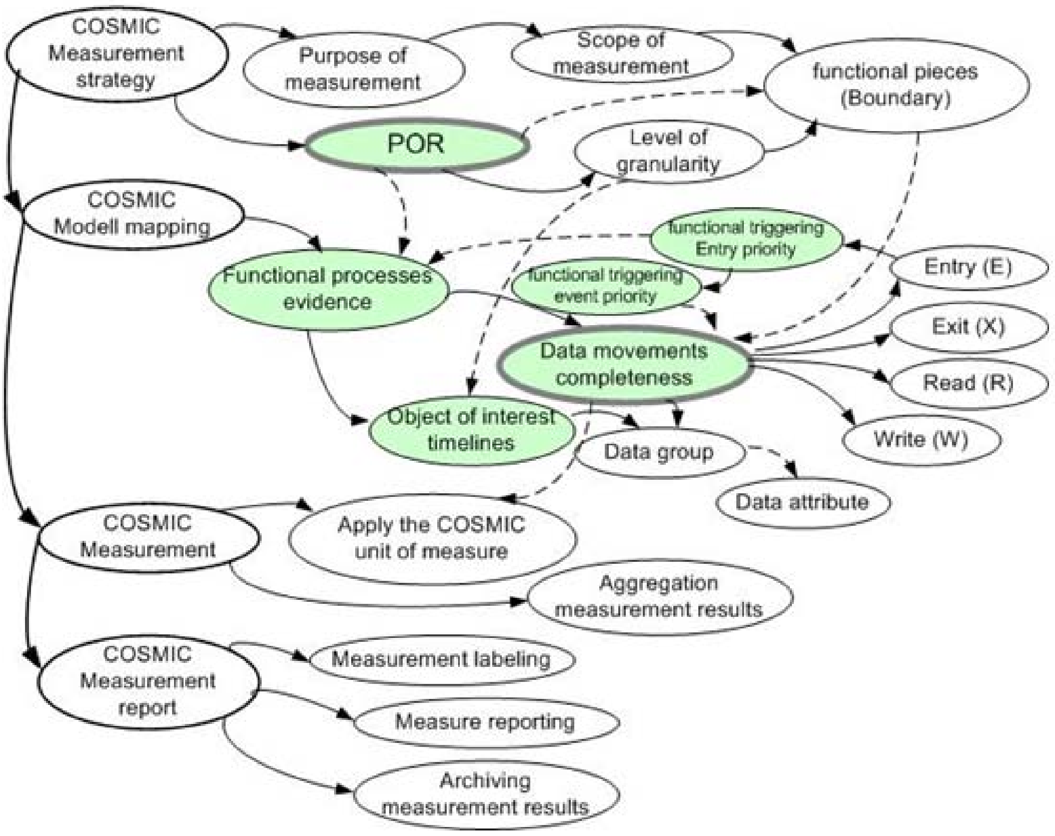
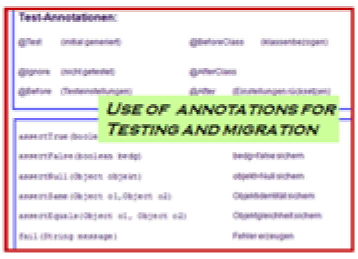

COSMIC Measurement strategy
Purpose of measurement
Scope of measurement
functional pieces (Boundary)
POR
Level of granularity
COSMIC Modell mapping
functional triggering Entry priority
Functional processes evidence
functional triggering event priority
Entry (E)
Exit (X)
Data movements completeness
Read (R)
Object of interest timelines
Data group
Write (W)
Data attribute
COSMIC Measurement
Apply the COSMIC unit of measure
Aggregation measurement results
COSMIC Measurement report
Measurement labeling
Measure reporting
Archiving measurement results

Figure 11: POR-extended COSMIC FP measurements

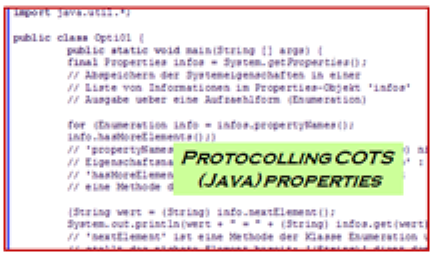
Examples of the organizational-process-based functionalities are characterized in the following figures Java examples.



Test-Annotationen:

```
@Test     (initial generiert)        @BeforeClass   (klassenbezogen)
@Ignore   (nicht getestet)           @AfterClass
@Before   (Testeinstellungen)        @After     (Einstellungen rücksetzen)

assertTrue(boole
assertFalse(boolean bedg)                  bedg=false sichern
assertNull(Object objekt)                  objekt=Null sichern
assertSame(Object o1,Object o2)            Objektidentität sichern
assertEquals(Object o1, Object o2)         Objektgleichheit sichern
fail(String message)                       Fehler erzeugen
```



```
import java.util.*;

public class Opti01 {
        public static void main(String [] args) {
        final Properties infos = System.getProperties();
        // Abspeichern der Systemeigenschaften in einer
        // Liste von Informationen im Properties-Objekt 'infos'
        // Ausgabe ueber eine Aufzaehlform (Enumeration)

        for (Enumeration info = infos.propertyNames();
        info.hasMoreElements();)
        // 'propertyNames
        // Eigenschaftsna
        // 'hasMoreElemen
        // eine Methode d

        {String wert = (String) info.nextElement();
        System.out.println(wert + " = " + (String) infos.get(wert
        // 'nextElement' ist eine Methode der Klasse Enumeration
```

POR maintainabiklity (as annotations for testing and migration in Java)

POR ressource characteristics (as property protocolling in Java)

Figure 12: POR Java examples

After this short characterization we will describe any aspects and principles of size measurement based om the COSMIC method.

4. COSMIC FP EXTENDED/MODIFIED MEASUREMENTS

4.1 Principles of COSMIC Extensions

In the COSMIC method 4.0 description we can found any principles of method extensions (as so-called local extension) like:

> "If it is judged necessary to account for complex algorithms, a local standard may be arranged for this exceptional functionality. In any functional process where there is an abnormally complex data manipulation functional sub-process, the Measurer is free to assign his or her own locally-determined Function Points" (p. 64)
>
> "When more precision is required in the measurement of data movements, then a sub-unit of the measure can be defined. For example, a meter can be sub-divided into 100 centimeters or 1000 millimeters. By analogy, the movement of a single data attribute could be used as a sub-unit of measurement. Measurements on a small sample of software in the field trials of COSMIC indicated that on the sample measured, the average number of data attributes per data movement did not vary much across the four types of data movement. " (p. 64)
>
> "Error/confirmation messages issued by the functional process being measured
> a) Identify one Exit to account for all types of error or confirmation messages issued by a functional process from all possible causes, e.g. success or failures of validation of entered data, or for a requirement to retrieve data or to make data persistent, or arising from the response from a service requested of another piece of software or intelligent hardware." (p. 57)

The extensions themselves could be realized by calibrations using additional CFPs depending on the "estimated" additional size. Considering the principles and rules of the COSMIC FP method, the following simple adaptations could be defined:

[COSMIC 2014], p. 45:

RULES – Entry (E) ***(QUR extended/modified)***
a) The data group of a ***real-time triggering Entry*** may consist of only one data attribute which simply informs the software that 'an event Y has occurred' . . .
b) . . .

[COSMIC 2014], p. 46:

RULES – Exit (X) ***(POR extended/modified)***
a) . . . Therefore, a single Exit shall be identified to represent all these message occurrences within each ***monthly functional process*** where they are required by the FUR.
b) . . .

[COSMIC 2014], p. 47:

RULES – Read (R) ***(QUR extended/modified)***
a) Identify a ***quality aspect of Read*** when, according to the FUR, the software being measured must retrieve a data group from persistent storage.
b) . . .

[COSMIC 2014], p. 47:

RULES –Write (W) ***(PUR extended/modified)***
a) Identify a ***cloud-based Write*** when, according to the FUR, the software being measured must move a data group to persistent storage.
b) . . .

In the same manner we could do any adaptations for the other internal elements of COSMIC measurements as object of interest, triggering event etc. in order to idenify only the QUR, PUR and POR pieces of size that is necessary to differ it from the original functionalities. In following we discuss two of examples of COSMIC extensions as general principles.

4.2 COSMIC SP Measurement

SP stands for *Software Product Points* in a general manner and should be measured the *size of the whole software system* or product. The simple execution of COSMIC SP as *CSP* can be described considering (9) to (12), (14) to (18), (21) to (24) and (26) as

$$CSP = CFP + COSMIC_{NUR}\ . \qquad (31)$$

where *NUR* (as non functional user requirements) summarizes the *QUR* and the *PUR* of the software product (as $NFR = NUR \cup POR$).

The $COSMIC_{NUR}$ requires the considerations of the CFP basic counters as (in a first simplified approximation we can assume that the aspects of *COMPL* are involved in the *QUR* and *PUR*)

$$E^{QUR}, X^{QUR}, R^{QUR} \text{ and } W^{QUR},$$

and

$$E^{PUR}, X^{PUR}, R^{PUR} \text{ and } W^{PUR}$$

in order to estimate/execute the different kind of product sizing as $SIZE_{artefact}$ and $SIZE_{empirical}$.

But, how we can count these empirical based *E, X, R* and *W*? Based on the COSMIC FP method, we need the following extensions and/or modifications:

(a) the *QUR* implies the *quality assurance process* that involves their own entries (E^{QUR}), exits (X^{QUR}), reads (R^{QUR}) and writes (W^{QUR}); examples of quality assurance processes are authorization procedures, user interface adaptation and input value controlling.

(b) the *PUR* implies the *platform ensuring process* that also involves their own entries (E^{PUR}) and exits (X^{PUR}), reads (R^{PUR}) and writes (W^{PUR}); examples of platform ensuring processes are platform emulation, performance controlling, infrastructure migration and component wrapping.

(c) the *entries* of the (given and described) COSMIC functional process could be quality- or platform-based (such like performance requirements) as E_{QUR} or E_{PUR},

(d) the *exits* of the COSMIC functional process could be quality- or platform-based as X_{QUR} or X_{PUR}.

(e) the *reads* of the COSMIC functional process could be quality- or platform-based (such like performance requirements) as R_{QUR} or R_{PUR},

(f) the *writes* of the COSMIC functional process could be quality- or platform-based as W_{QUR} or W_{PUR}.

The extension of the COSMIC FP method by introducing quality assurance processes and platform ensuring processes (as cases *(a)* and *(b)*) is a simple adaptation of the existing principles and rules for these further considered processes. Therefore, a simple example of COSMIC SP measurement could be

$$CSP = CFP + \#(E^{NUR}, X^{NUR}) + |\{R^{NUR}, W^{NUR}\}| \qquad (32)$$

The cases *(c)* to *(f)* need an introduction of empirical evaluations (like in the IFPUG FP method). But, it should be conform to the current software system characteristics and modern paradigms.

Hence, we suggest (based on our experience) the following evaluations as an *initial calibration* ([Kunz 2007], [Schmietendorf 2013], [Schmietendorf 2010])

- considering service oriented systems (SOA) we derived the following quality sizing for chosen (industrial) SOA systems

$$COSMIC_{NUR} = \#(E_{QUR}^{SOA}, X_{QUR}^{SOA}) \approx 0.4\ CFP \qquad (33)$$

- considering cloud computing application we suggest the following platform sizing for chosen software application in cloud computing

$$COSMIC_{NUR} = \#(E_{PUR}^{Cloud}, X_{PUR}^{Cloud}, R_{PUR}^{Cloud}, W_{PUR}^{Cloud}) \approx 0.25\ CFP \qquad (34)$$

Note, that these results are based on special software consideration and need more experience for the general applicability. But, we only show the principles of CFP extensions in the described manner.

In this way, we obtain the size of the whole software system as *total product size measurement* that can be used for comparison with other whole system sizing methods.

4.3 COSMIC PP Measurement

PP stands for *Software Process Points* in a general manner and should be measured *the size of software processes* like development, maintenance or application. The simple execution of COSMIC PP as *CPP* can be described considering (9) and (13), (19) to (20), (24) and (26) as

$$CPP = COSMIC_{POR}\ . \qquad (35)$$

where *POR* we have defined above as a set as {*development method, life cycle, management aspects, personal resources, CASE tools, COTS, hardware resources*} and *management aspects* = {*timeline, effort, costs, size*}. Note, that this list is a typical conclusion from our given experience and our references and can be differ in any other environment or IT areas.

The $COSMIC_{POR}$ requires the considerations of the *CFP* basic counters as

$E^{POR}, X^{POR}, R^{POR}$ and W^{POR}

in order to estimate/execute the different kind of product sizing as $SIZE_{process}$.

In the same manner like the *QUR*- and *PUR*-based SP counting, we can define the following *POR*-based modifications for PP counting as:

(g) the *POR* implies the *project organizational process* that involves their own entries (E^{POR}), exits (X^{POR}), reads (R^{POR}) and writes (W^{POR}),

(h) the *entries* of the (given and described) COSMIC functional process could be quality- or platform-based (such like performance requirements) as E_{POR},

(i) the *exits* of the COSMIC functional process could be quality- or platform-based as X_{POR}.

(j) the *reads* of the COSMIC functional process could be quality- or platform-based (such like performance requirements) as R_{POR},

(k) the *writes* of the COSMIC functional process could be quality- or platform-based as W_{POR}.

Here again, the extension of the COSMIC FP method with *(g)* is a simple adaptation such as

$$CPP = \#(E^{POR}, X^{POR}) + |\{R^{POR}, W^{POR}\}| \qquad (36)$$

In the cases of *(h)* to *(k)* we must define any initial empirical evaluations. As an example we will chose the *application process* involving the size aspects of system handling and (simple) controlling (as control entries and exits etc.) with the following evaluations ([Schmietendorf 2012], [Schmietendorf 2007], [Wille 2011]):

- considering service oriented systems (SOA) we derived the following application process sizing for chosen (industrial) SOA systems

$$CPP = \#(E^{SOA}_{POR}, X^{SOA}_{POR}) \approx 0.14\ CFP \qquad (37)$$

- considering cloud computing application we suggest the following application process sizing for chosen software application in cloud computing

$$CPP = \#(E^{Cloud}_{POR}, X^{Cloud}_{POR}, R^{Cloud}_{POR}, W^{Cloud}_{POR}) \approx 0.2\ CFP\,. \qquad (38)$$

In this way, we obtain the *size of the software process(es)* as a essential basis for software management.

4.4 COSMIC Extensions Applications

The essential areas of COSMIC extension application can be characterized as following (using (31) and (35)):

- The deriving of the whole software system size as *CSP* allows to estimate *development, maintenance and application effor*t of a software product *SP* in the following manner:

$$effort_{development}(SP) = \alpha\,(CSP+CPP_{development})\ [PM] \qquad (39)$$

where $\alpha \approx 0.05$ for our software examples, because the extended sizing considers the effort basis of the *NFR* themselves,

$$effort_{maintanance}(SP) = \beta\,(CSP+CPP_{maintenance})\ [PM] \qquad (40)$$

where $\beta \approx 0.2\alpha$ using the experience of the ISBSG data above.

$$effort_{application}(SP) = \gamma\, CPP_{application}\ [PM] \qquad (41)$$

where $\gamma \approx 0.014$ for our examples of measured software systems.

- The consideration of the *NFR*-based allows measures the size and estimates the effort of the quality assurance process and the platform ensuring process separately.

- The deriving of the software processes size as *CPP* allows to compare different kinds of software processes as waterfall, evolutionary or agile development.

- Other applications of the COSMIC FP extensions could be used for classifying different software system by their complexity and manageability considering the $COMPL_{artefact} \oplus COMPL_{empirical}$ relationships.

5. FURTHER COSMIC MEASUREMENTS

5.1 General Intentions of other COSMIC Measurements

Note that the measurements are addressed mainly to the requirements and not to source code, manuals, test scripts or something like that. The following figure shows any aspects of these further measurements.

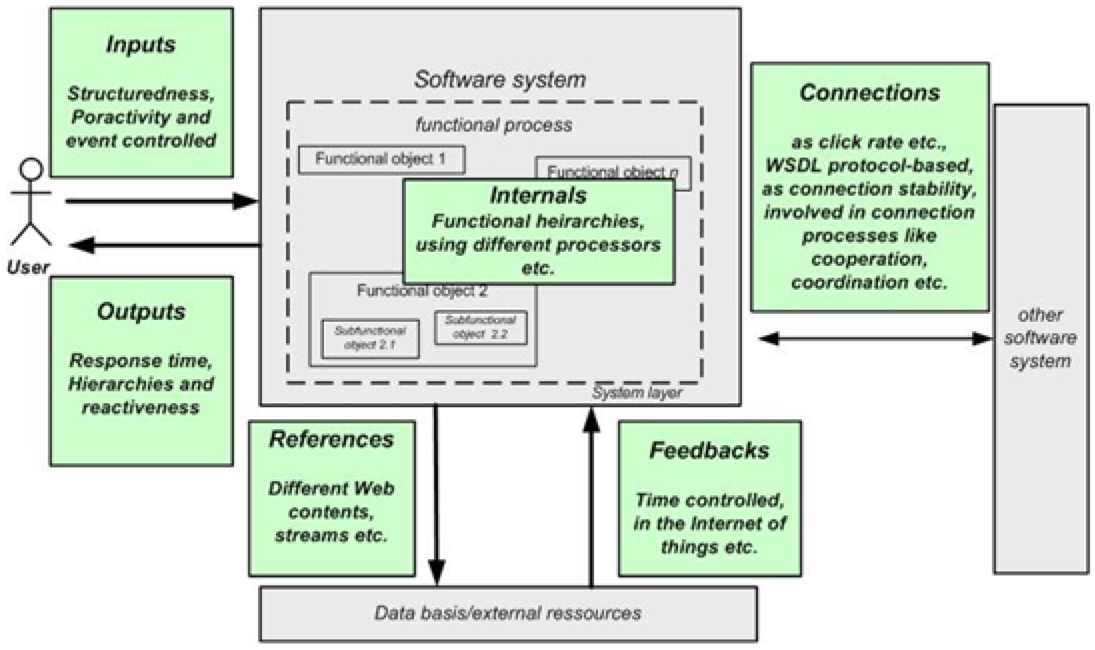

Inputs
Structuredness, Poractivity and event controlled
User
Outputs
Response time, Hierarchies and reactiveness
Software system
functional process
Functional object 1
Functional object n
Internals
Functional heirarchies, using different processors etc.
Functional object 2
Subfunctional object 2.1
Subfunctional object 2.2
System layer
Connections
as click rate etc., WSDL protocol-based, as connection stability, involved in connection processes like cooperation, coordination etc.
other software system
References
Different Web contents, streams etc.
Feedbacks
Time controlled, in the Internet of things etc.
Data basis/external ressources

Figure 13: Other kinds of software measurements

Therefore, we can/should consider more than the system I/O charcateristics as E, X, R and W such as

- considering the complexity of the E, X, R, W themselves involving their set and structure of attributes,
- general input/output description with detailed references, relationships and other process ingridients like proactivity, self controlling etc.,
- more internal characteristics of functionalities and functional processes in the systems themselves.

5.2 COSMIC CP Measurement

CP stands for *Software Complexity Points* as measurement of the complexity as indicator for usability, effort and comprehension (see COMPL characterizations in (21) to (23)). Therefore, a general classification of system complexity is characterized by Lehmann (see [Pfleeger 1998]) as

- *S systems*: *as simple system* including well-defined algorithms and programming techniques
- *P systems*: *as partial algorithmic-based system* including non deterministic algorihms solved by any interactive solutions and paradigms
- *E systems*: *as extreme system out of algorithmic in general* could be given in country-based ecosystems etc.

Another general overview about the main types of software complexity is defined by Jones [Jones 2007] as annotations in the following class diagram.

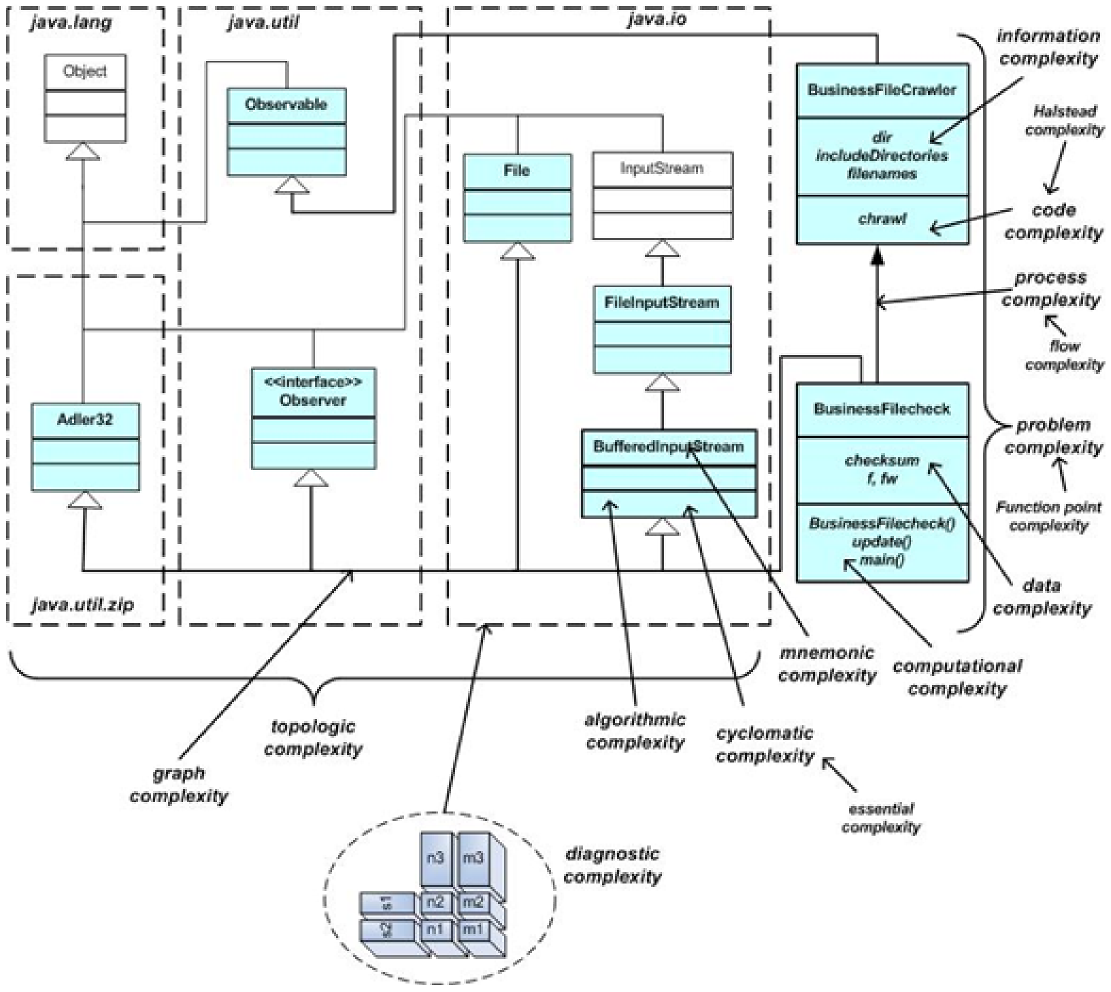


Figure 14: Complexities of Jones charcterized in a Java class diagram

Two of them are described in more details in the following general characterizations [Jones 2007]

(1) *Data complexity:* “deals with the number of attributes associated with entities.” Its importance has been increased using complex network technologies like *Grid or Cloud Computing* (see [Agapi 2011], [Bhowmick 2004], [Jatuun 2009], [Papazoglou 2011], [Rud 2006], [Schmietendorf 2007] and [Yau 2011]).

(2) *Flow complexity:* “is a major topic in the studies of fluid dynamics and meteorology. It deals with the turbulence of fluids moving through channels and across obstacles.” This complexity could be used for characterization the *Web service application and orchestration* (see [Ahn 2011], [Armbrust 2010], [Banerjee 2011], [Dumke 2008], [Neumann 2013], [Sing 2005] and [Wei 2010]).

Adapting the basics of the COSMIC FP method, we must explain the meaningfulness of the considerations as

$$E^{COMPL}, X^{COMPL}, R^{COMPL} \text{ and } W^{COMPL}.$$

Note, the number of attributes/parameters of an exit or entry is not considered in the original COSMIC method. Therefore, a very simple extension could define in the counting of these aspects. This leads to the simple characterization as

$$E_{data}^{COMPL} = \#attributes(E), \qquad X_{data}^{COMPL} = \#attributes(X), \tag{42}$$

and

$$R_{data}^{COMPL} = \#parameters(R), \qquad W_{data}^{COMPL} = \#parameters(W). \tag{43}$$

Furthermore, we can consider the measurement aspects of flow complexity like

$$E_{flow}^{COMPL},\ X_{flow}^{COMPL},\ R_{flow}^{COMPL} \text{ and } W_{flow}^{COMPL}.$$

The entries and exits could be services (as flow process indicators). Thererfore, we can apply the Rud service complexity metrics like [Rud 2006]

- $\mu_{NSIC}^{complexity}$ *as* number of services involved in the compound service that increase the complexity in the manner of structured depenedencies and
- $\mu_{SIY}^{complexity}$ as number of independed services in the system that expresses a descreasing of complexity because of lower dependencies.

These metrics lead to any extensions of the COSMIC data movements considering the *relationships between the Entries and Exits* written as pair in following

$$\#(E_{flow}^{COMPL},\ X_{flow}^{COMPL}.) \tag{44}$$

and in details as

$$\#(\ \{\{E_i\} \mid E_i \in E_{compound}\}_{NSIC}^{COMPL}\ ,\ \{\{X_j, X_k\} \mid X_j \cap X_k = \emptyset\}_{SIY}^{COMPL}\ ) \tag{45}$$

The reads and writes could be service application based on (persitstent stored) data/service basis. Thererfore, we can apply the Rud service complexity metrics like [Rud 2006]

- $\mu_{CVS}^{complexity}$ *as* count of simultaneous versions of the service that increases the complexity in the manner of service variability and
- $\mu_{MCFS}^{complexity}$ as metadata (*md*) change frequency of the service that increases the complexity because of higher changements of service descriptions.

These metrics lead to any extensions of the COSMIC data movements considering the *changements in the Reads and Writes* written as pair in following

$$\#(R_{flow}^{COMPL},\ W_{flow}^{COMPL}.) \tag{46}$$

and in details as

$$\#(\ \{\{R\} \mid t(R_i) = t(R_j)\}_{CVS}^{COMPL}\ ,\ \{\{W_k\} \mid \Delta t(mdW_k)\}_{MCFS}^{COMPL}\ ) \tag{47}$$

In order to achieve more evidence in industrial applications, the characteristics of the *Cloud computing, Internet of Things* and *Big data* could be involved in these considerations (see [Fiegler 2014]. [Nair 2015] and [Neumann 2013]).

5.3 COSMIC DP Measurement

DP stands for *Software Document Points* as the further essential part of software systems and infarstructures. There are different kinds of software documents that can be found in the different software processes like (see [Andersson 2006], [Jones 2010], [Kandt 2006], [Pfleeger 1998], [Sommerville 2010])

- *Software development documents*: Description and charts of the software models, architectures and implementations like formal specifications, UML diagrams, test cases documentation, program comments etc.,

- *Software maintenance documents*: Trouble reports, test scenarious, configuration descriptions etc.

- *Software application documents*: User manuals, reference documentations, help documentations etc.

Document measurements are given in the following kinds and intentions (see [Hobelsberger 2012], [Laird 2006], [Lehner 1994], [Mencke 2010])

(a) Software document measurements: considering of the measures like readability, understandability, changeability, document sizes etc.

(b) Web ressources measurements: identifying the counts of ressources sizes, performance, stability, availability, frequency etc.

(c) Documents as Web contents measurements: considering of the operationalities, process involvements, content quality, usability, completeness etc.

(d) Social network measurements: determination of user behavior, size of communities, size of user groups, user frequency etc.

Adapting the COSMIC method for document measurements, we must define any measurement princplies and rules as

$$E^{DOC}, X^{DOC}, R^{DOC} \text{ and } W^{DOC}.$$

A first simple definition of *DP* could be the counting of the consideration of the data movements in the software docuements like a user manual characterized as

$$\#(\{\{E^{DOC}\} | E_i^{DOC} \cong E_i\}, \{\{X^{DOC}\} | X_j^{DOC} \cong X_j\}, \{\{R^{DOC}\} | R_k^{DOC} \cong R_k\}, \{\{W^{DOC}\} | W_l^{DOC} \cong W_l\},) \qquad (48)$$

Where is described which elements of functional process as I/O characteristics are involved in the documentation and which are not.

Further descriptions of document measurements or measurements of the documentation could be based on the characterizations in (a) to (d) and are intended in the same manner like (48).

6. CONCLUSION AND FUTURE WORK

This paper discusses the extension of the COSMIC FP method considering the non functional requirements with their modification of the basis counters (as *E, X, R* and *W*) and involving further *NFR*-based processes for complete deriving of software sizes.

On the one hand, the COSMIC extensions consider the empirical aspects of the I/O counting of the COSMIC FP method by scaling the input/output counters themselves.

On the other hand, the COSMIC extensions introduce empirical based processes (quality assurance and platform ensuring processes) as a extended functional processes that would be measured as the same manner like the I/O counting in the COSMIC FP method itself.

This point of view qualifies the COSMIC-based effort estimation *from the black-box estimation* as

$$effort_{CFP\text{-}based} = \alpha_{POR}^{QUR,PUR} \times size_{CFP-based}^{FUR}$$

to the *white-box estimation* as

$$effort_{CFP\text{-}based} = \alpha \times [size_{CFP-based}^{POR} + size_{CFP-based}^{QUR} + size_{CFP-based}^{PUR} + size_{CFP-based}^{FUR}]$$

Two examples arre the extended measurement-based formulas as *COSMIC function points* involving *COSMIC product point* and *COSMIC process points* as

COSMIC software size = $CSP^{CFP} + CPP$

with ($NUR = QUR \oplus PUR$)

$$CSP = CFP + \#(E^{NUR}, X^{NUR}) + |\{R^{NUR}, W^{NUR}\}|$$

and

$$CPP = \#(E^{POR}, X^{POR}) + |\{R^{POR}, W^{POR}\}|$$

The other kinds of extensions are the application of the COSMIC method for measurements of other characteristics/attribtes of software like *complexity* and *documentation.* Examples of these extensions are

COSMIC complexity points as

$$CCP = \#(E^{COMPL}, X^{COMPL}, R^{COMPL}, W^{COMPL})$$

COSMIC documentation points as

$$CDP = \#(E^{DOC}, X^{DOC}, R^{DOC}, W^{DOC})$$

This papers discussed the general principles of COSMIC extemsions. But, the detailed principles and rules must be defined in next steps. Furthermore, experience in the industrial sector for the software system sizing in order to achieve more granularity and refinements in the software product and process measurement based on the kernel idea of the COSMIC FP method are necessary.

REFERENCES

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

[Agapi 2011] Agapi, A. et al.: *Routers for the Cloud.* IEEE Internet Computing, Sept./Oct. 2011, pp. 72-76

[Ahn 2011] Ahn, G.; Shebab, M.; Squicciarini, A.: *Security and Privacy in Social Networks.* IEEE Internet Computing, May/June 2011, pp. 10-12

[Armbrust 2010] Arnbrust, M. et al.: *A View of Cloud Computing.* Comm. of the ACM, 53(2010)4, pp. 50-58

[Andersson 2006] Andersson, E.; Greespun, P.; Grumet, A.: *Software Engineering for Internet Applications.* MIT Press, Cambridge, Mass., 2006

[Banerjee 2011] Banerjee, P. et al.: *Everything as a Service: Powering the New Information Economy.* IEEE Computer, March 2011, pp. 36-43

[Bhowmick 2004] Bhowmick, S. S.; Madria, S. K.; Ng, W. K.: "Web Data Management*."* Srpinger Publ., 2004

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

[Bundschuh 2008] M. Bundschuh, C. Dekkers, "The IT Measurement Compendium" Springer Publ., 2008

[Chemutui 2009] M. Chemuturi, "Software Estimation Best Practices" Tools & Techniques. J. Ross Publ., Lauderdale, FL, 2009

[COSMIC 2014] COSMIC-FFP: Measurement Manual – The COSMIC Implementation Guide for ISO/IEC 19761:2014. v 4.0, (http://www.cosmicon.com, 2014)

[Dumke 2010] R. Dumke, A. Abran, "COSMIC Function Points: Theory and Advanced Practices" CRC Press, Boca Raton, 2010

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

[Dumke 2014] R. Dumke, A. Schmietendorf, M. Seufert, C. Wille, „Handbuch der Softwareumfangsmessung und Aufwandschätzung." Logos-Verlag Berlin, 2014

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

[Fiegler 2014] Fiegler, A.; Zwanziger, A.; Herden, S.; Zenker, N.; Dumke, R.: *Analyse von Ressource Pooling Varianten in Cloud-Systemen.* In: Büren et al.: Praxis der Software-Messung, Shaker-Verlag, Aachen, 2014, S. 53-66

[ISBSG 2012] ISBSG Software Project Estimation – A Workbook for Macro-Estimation of Software Development Effort and Duration. Melbourne, 2003 (Release 12, 2012, http://www.isbsg.org)

[Jatuun 2009] Jaatun et al.: "Cloud Computing*."* LNCS 5931, Springer Publ., 2009

[Jones 2007] C. Jones, "Estimating and Measuring Software Costs: Bringing Realism to Estimating" McGraw-Hill Publ., 2007

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

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

[Kunz 2007] M. Kunz, R. Dumke, "Empirical Foundations of COSMIC FFP Application for Effort Estimation" (German). Preprint No 7, University of Magdeburg, 2007 (http://www-ivs.cs.uni-magdeburg.de/sw-eng/agruppe/ forschung/Preprints.shtml)

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

[Lehner 1994] Lehner, F.: "Messung der Software-Dokumentation und Messung der Dokumentationsqualität." Carl-Hanser-Veröag, München, 1994

[Leiss 2007] E. L.Leiss, "A Programmer’s Companion to Algorithm Analysis" Chapman & Hall Publ., 2007

[Lother 2001] M. Lother, R. Dumke, "Point Metrics – Comparison and Analysis" In Dumke/Abran: Current Trends in Software Measurement, Shaker Publ., Aachen, 2001, pp. 228-267

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

[Nair 2015] Nair, R.: “Big Data Needs Approximate Computing”, CACM, 58(2015)1, P. 104

[Neumann 2013] Neumann, R.: "The Internet of Products." Springer Publ., Berlin, New York , 2013

[Papazoglou 2011] Papazoglou, M. P.; Andrikopoulos, V.; Benbernou, S.: *Managing Evolving Services.* IEEE Software, May/June 2011, pp. 49-55

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

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

[Schmietendorf 2012] A. Schmietendorf, "Kosten- und Aufwandschätzung bei der Entwicklung von Smartphone-Apps, eine Bestandsaufnahme" Büren, G.; Dumke, R.; Ebert, C.; Münch, J.: MetriKon 2012 - Praxis der Software-Messung. Shaker Publ., Aachen, 2012, pp. 141-148

[Schmietendorf 2007] A. Schmietendorf, R. Dumke, "Approaches of Effort Estimation in the Age of Agile Development" (German). In: Büren et. al.: Practice of Software Measurement, Shaker Publ., Aachen, 2007, pp. 309-326

[Schmietendorf 2013] A. Schmietendorf, A. Fiegler, R. Neumann, C. Wille, R. R. Dumke, "COSMIC Functional Size Measurement of Cloud Systems" Proc. of the IWSM 2013, October, 24 - 26, 2013, Ankara, Turkey

[Schmietendorf 2010] A. Schmietendorf, R. Neumann, R. Dumke, "COSMIC and SOA Sizing - A critical analysis and proposals for improvement potentials" In: Abran et al.: Applied Software Measurement, Shaker-Verlag, Aachen, 2010, S. 559-569

[Singh 2005] Singh, M. P.: "The Practical Handbook of Internet Computing." Chapman & Hall CRC Publ., 2005

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

[Sommerville 2010] Sommerville, I., "Software Engineering." Addison-Wesley, 11th edition, 2010

[Symons 2013] C. Symons, "Guideline for Measurement Strategy Patterns" Version 1., March 2013

[Wei 2010] Wei Y.; Blake, M. B.: *Service-Oriented Computing and Cloud Computing – Challenges and Opportunities.* IEEE Internet Computing, Nov./Dec. 2010, pp. 72-75

[Wille 2011] C. Wille, A. Fiegler, R. Neumann, R. R. Dumke, "Evidence-Based Evaluation of Effort Estimation Methods" Proceedings of the IWSM-MENSURA 2011, November 3-4, 2011, Nara, Japan, IEEE Computer Society Los Alamitos, California, Washington, Tokyo, S. 196-208

[Yau 2011] Yau, S. S.; An, H. G.: *Software Engineering Meets Service and Cloud Computing.* IEEE Computer, October 2011, pp. 47-53

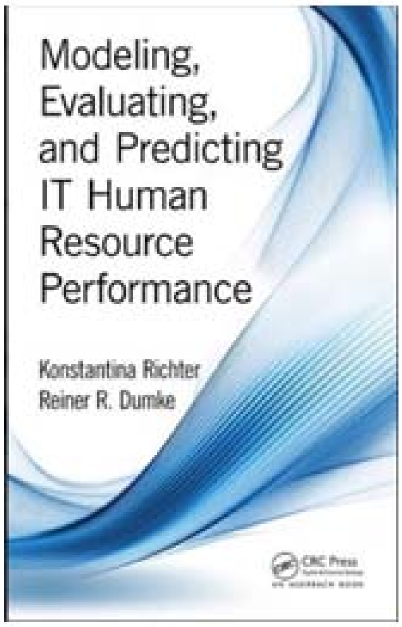
[Zuse 1998] H. Zuse, “A Framework of Software Measurement”, DeGruyter Verlag, Berlin, 1998

Konstantina Richter, Reiner Dumke:

Modeling, Evaluating and Predicting IT Human Resource Performance

CRC Press, Boca Raton, Florida, 2015 (275 Seiten)

Modeling, Evaluating, and Predicting IT Human Resources Performance explains why it is essential to account for the human factor when determining the various risks in the software engineering process. The book presents an IT human resources evaluation approach that is rooted in existing research and describes how to enhance existing approaches through strict use of software measurement and statistical principles and criteria.


Modeling, Evaluating, and Predicting IT Human Resource Performance
Konstantina Richter
Reiner R. Dumke
CRC Press

Büren, G.; Dumke, R.R.; Ebert, C, Münch, J., Seufert, M.:

MetriKon 2014 - Praxis der Softwaremessung
Tagungsband des DASMA Software Metrik Kongresses
6. - 7. November 2014, Stuttgart

Shaker Verlag, Aachen, 2014 (222 Seiten)

The book includes the proceedings of the MetriKon 2014 held in Stuttgart in November 2014, which constitute a collection of theoretical studies in the field of software measurement and case reports on the application of software metrics in companies and universities.

Magdeburger Schriften zum Empirischen Software Engineering

Hrsg: [illegible]

OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG

Fakultät für Informatik

Institut für Verteilte Systeme

Arbeitsgruppe Softwaretechnik

MetriKon 2014

Praxis der Software-Messung

Tagungsband des DASMA Software Metrik Kongresses

6.-7. November 2014

Vector Consulting Services, Stuttgart

[illegible]

Vogelezang, F., Daneva, M.:

IWSM-MENSURA 2014 Proceedings
October 6 - 8, 2014, Rotterdam, Netherlands

IEEE CPS Publishing Service (online), 2014

This proceedings includes the full papers and the short papers of the 2014 Conference of the 24nd International Workshop on Software Measurement (IWSM) and the 2014 Ninth International Conference on Software Process and Product Measurement (MENSURA).

IWSM MENSURA 2014 - ROTTERDAM

Embarking for new developments in software measurement

Proceedings of the joint conference of the 24th International Workshop on Software Measurement (IWSM) and the 9th International Conference on Software Process and Product Measurement (Mensura)

6-8 October 2014 - Rotterdam, the Netherlands

Frank Vogelezang & Maya Daneva (eds.)

nesma

Published by CPS

Qualitative und quantitative Bewertungsaspekte bei der agilen Softwareentwicklung plattformübergreifender mobiler Applikationen

André Nitze, Andreas Schmietendorf

2014, 168 Seiten
ISBN 978-3-8325-3774-6
Preis: 24.00 €

Der Themenschwerpunkt der vorliegenden Monografie beschäftigt sich mit der professionellen Entwicklung und Bereitstellung mobiler Business-Apps, die im Kontext unternehmerischer Aktivitäten zum Einsatz kommen. Bei der Softwareentwicklung gilt es, vielfältigen Qualitätsanforderungen wie z.B. der Performance, der Wartbarkeit, der Plattformunabhängigkeit, der Ergonomie oder der Sicherheit gerecht zu werden. Darüber hinaus bedarf es des Managements betrieblich eingesetzter Geräte und Apps unter Berücksichtigung unternehmensspezifisch festzulegender Mobilitätsstrategien. Innerhalb der vorliegenden Forschungsarbeit wurden neben diesen Themen auch spezielle Aspekte wie der datenschutzrechtliche Umgang mit Nutzerdaten oder auch Qualitätsmodelle und Ansätze zur plattformübergreifenden Entwicklung behandelt.

Der Themenschwerpunkt der vorliegenden Monografie beschäftigt sich mit der professionellen Entwicklung und Bereitstellung mobiler Business-Apps, die im Kontext unternehmerischer Aktivitäten zum Einsatz kommen. Bei der Softwareentwicklung gilt es, vielfältigen Qualitätsanforderungen wie z.B. der Performance, der Wartbarkeit, der Plattformunabhängigkeit, der Ergonomie oder der Sicherheit gerecht zu werden. Darüber hinaus bedarf es des Managements betrieblich eingesetzter Geräte und Apps unter Berücksichtigung unternehmensspezifisch festzulegender Mobilitätsstrategien. Innerhalb der vorliegenden Forschungsarbeit wurden neben diesen Themen auch spezielle Aspekte wie der datenschutzrechtliche Umgang mit Nutzerdaten oder auch Qualitätsmodelle und Ansätze zur plattformübergreifenden Entwicklung behandelt.

Bestellung ¨uber den Buchhandel oder direkt beim Verlag, entweder online oder per Fax beim Logos Verlag Berlin GmbH· Comeniushof – Gubener Str. 47 · D-10243 Berlin

Schmietendorf, A. (Hrsg.):

Eine praxisorientierte Bewertung von Architekturen und Techniken für Big Data

(110 Seiten) Shaker-Verlag Aachen, März 2015 ISBN 978-3-8440-2939-0

Hochschule für
Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Berliner Schriften zu modernen Integrationsarchitekturen
Hrsg.: Prof. Dr.-Ing. habil. Andreas Schmietendorf

Hochschule für Wirtschaft und Recht Berlin
Fachbereich II
Wirtschaftsinformatik – Systementwicklung

Eine praxisorientierte Bewertung von Architekturen und Techniken für Big Data

SHAKER VERLAG

Die Idee zum vorliegenden Buch entstand während der Durchführung von Seminaren, Workshops und Lehrveranstaltungen zum Thema Big Data. Die sowohl im industriellen als auch universitären Umfeld durchgeführten Veranstaltungen verdeutlichten den Bedarf einer praxisorientierten Auseinandersetzung mit den vielfältig angebotenen Architekturansätzen und Techniken. Mit Hilfe des Buchs soll dem entsprechend eine Einarbeitung in das sich ständig verändernde Big Data Ökosystem unterstützt werden. Dabei geht es weniger um eine Favorisierung nur eines Frameworks als vielmehr um die Anregung einer kritischen Auseinandersetzung mit alternativen Systemlösungen. Neben der Verdeutlichung von Einsatzszenarien galt das besondere Interesse den mannigfaltigen Integrations- und Migrationsanforderungen einer realen Big Data Lösung. Die Möglichkeiten zur Berücksichtigung vielfältiger Datenquellen und Persistenzmechanismen haben maßgeblichen Einfluss auf den Erfolg entsprechender Big Data Ansätze. Der einführende Beitrag beschäftigt sich mit grundlegenden Eigenschaften von Big Data Lösungen und möglichen Systemansätzen. Darauf aufbauend geht ein weiterer Beitrag auf die technischen Details des Apache Hadoop-Kerns und die detaillierte Abbildung des MapReduce-Algorithmus ein. Die Architektur und Einsatzmöglichkeiten von NoSQL-Datenbanksystemen stehen im Mittelpunkt der folgenden Beiträge. Im Einzelnen werden Apache HBase, MongoDB sowie der zur echtzeitbasierten Suche einsetzbare ElasticSearch-Ansatz aufgegriffen. Mit SAP Hana existiert ein alternativer Architekturansatz für das SAP-Umfeld. Neben einer bodenständigen Einordnung und Abgrenzung zu klassischen BI-Ansätzen geht es im Beitrag insbesondere um mögliche Einsatzszenarien und Aspekte der Migration. Der abschließende Beitrag greift im Sinne eines Exkurses die cloudbasierte Bereitstellung einer Hadoop-Installation mit Hilfe der Container-Technologie Docker auf. Dabei wird unter anderem auf die Apache Ambari Lösung eingegangen, welche zur Bereitstellung, Konfiguration und Überwachung eines Hadoop Clusters verwendet werden kann. Mit dem vorliegenden Buch soll eine ingenieurmäßige Auseinandersetzung mit den aktuellen Big Data Technologien angeregt werden, dem entsprechend würde sich die Autoren über Feedbacks und weiterführende Diskussionen freuen. Für die konstruktive Zusammenarbeit möchte ich mich bei allen Autoren bedanken. Ebenso bei Frau Leany Maaßen vom Shaker Verlag Aachen für ihre schnelle und unkonventionelle Unterstützung des Buchprojekts.

Christof Ebert:

Risikomanagement kompakt

- Risiken und Unsicherheiten bewerten und beherrschen

Springer-Verlag, 2014, ISBN 978-3-642-41047-5

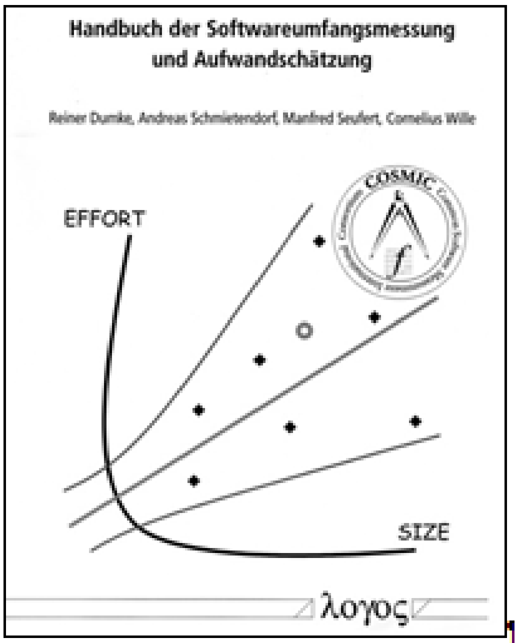
Risikomanagement ist das Schlüsselwerkzeug für Führungskräfte im Projekt und in der Linie. Es hilft dabei, Chancen, Unsicherheiten und Gefahren bewusst und proaktiv anzupacken, und damit kritische Probleme zu vermeiden. Sein pragmatischer Einsatz ist heute überlebensnotwendig und aufgrund von wachsenden Anforderungen an Produkthaftung und Governance für die Unternehmensführung verpflichtend. Das deutschsprachige Standardwerk "Risikomanagement kompakt" ist jetzt in einer komplett überarbeiteten neuen Auflage bei Springer erschienen. Das Buch fasst praxisnah zusammen, was Risikomanagement ist, wie es eingeführt und eingesetzt wird.

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

Den Kern des Buches bildet eine erstmals umfassende und vollständige Beschreibung einer exakten Bestimmung des Softwarefunktionsumfangs nach den so genannten COSMIC Function Points. Dabei werden neben der Methode selbst auch umfassende Beispiele für die verschiedensten Anwendungsgebiete und -paradigmen, wie Business Applikationen, SOA, Cloud Computing, wissenschaftlich-technische Berechnungen und vor allem auch für eingebettete Systeme ausführlich dargestellt, die auch die Grundlage für eine mögliche Zertifizierung nach dieser Methode bilden. Für die Anwendung dieser Methode werden einige Tools und Web-Dienste vorgestellt. Ebenso wird die Relevanz und der Inhalt einer internationalen Erfahrungsdatenbasis zur Aufwandschätzung erläutert.



Handbuch der Softwareumfangsmessung und Aufwandschätzung

Reiner Dumke, Andreas Schmietendorf, Manfred Seufert, Cornelius Wille

Schmietendorf, A.; Simon, F.:

BSOA/BCloud 2014

9. Workshop Bewertungsaspekte serviceorientierter Architekturen

4. November 2014, Frankfurt

Shaker Verlag, Aachen, 2014 (112 Seiten), ISBN 978-3-8440-2108-0

The book includes the proceedings of the BSOA/BCloud 2014 held in Frankfurt in November 2014, which constitute a collection of theoretical studies in the field of measurement and evaluation of service oriented and cloud architectures.

Hochschule für Wirtschaft und Recht Berlin
Berlin School of Economics and Law

Berliner Schriften zu modernen Integrationsarchitekturen
Hrsg.: Andreas Schmietendorf, HWR Berlin
Frank Simon, BLUECARAT AG Köln

Hochschule für Wirtschaft und Recht Berlin
Fachbereich II
Wirtschaftsinformatik – Systementwicklung

BSOA/BCloud 2014

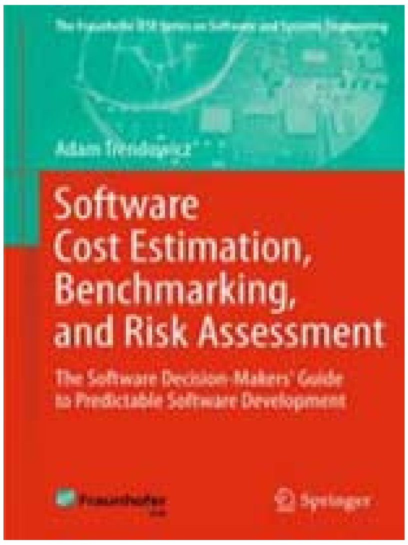
9. Workshop Bewertungsaspekte service- und cloudbasierter Architekturen

04. November 2014, Frankfurt/Main

Adam Trendowicz;

Software Cost Estimation, Benchmarking, and Risk Assessment - The Software Decision-Makers

Springer-Verlag, 2013, ISBN: 978-3-642-30763-8


The Fraunhofer IESE Series on Software and Systems Engineering
Adam Trendowicz
Software Cost Estimation, Benchmarking, and Risk Assessment
The Software Decision-Makers' Guide to Predictable Software Development
Fraunhofer
Springer

Richard Seidl und Harry Sneed:

Softwareevolution

dpunkt-Verlag, 2013

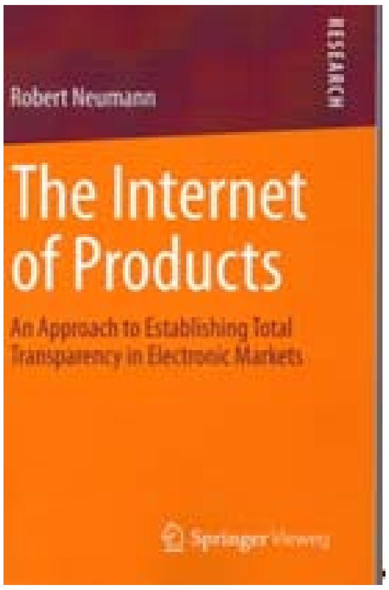
Harry M. Sneed · Richard Seidl
Software-evolution
Erhaltung und Fortschreibung bestehender Softwaresysteme
dpunkt.verlag

Robert Neumann:

The Internet of Products

An Approach to Establishing Total Transparency in Electronic Markets

Springer Vieweg, 2013 (263 Seiten), ISBN: 978-3-658-00904-5



RESEARCH

Robert Neumann

The Internet of Products

An Approach to Establishing Total Transparency in Electronic Markets

Springer Vieweg

Janus, A.:

Konzepte für Agile Qualitätssicherung und -bewertung in Wartungs- und Weiterentwicklungs-Projekten

Shaker Verlag, 2013 (177 Seiten), ISBN: 978-3-8440-1578-2

Magdeburger Schriften zum Empirischen Software Engineering

[illegible]

OTTO-VON-GUERICKE-UNIVERSITÄT MAGDEBURG

Fakultät für Informatik
Institut für Verteilte Systeme
Arbeitsgruppe Softwaretechnik

André Janus

Konzepte für Agile Qualitätssicherung und -bewertung in Wartungs- und Weiterentwicklungs-Projekten

[illegible]

SHAKER

Software Measurement Involved Conferences

January 2015:

SWQD 2015:	**Software Quality Days** January 20-22, 2015, Vienna, Austria see: http://2015.software-quality-days.com/en/conference/overview
ICPE 2015:	**5th ACM/SPEC International Conference on Performance Engineering** January 31- February 4, 2015, Austin, Texas, USA see: http://icpe2015.ipd.kit.edu/

February 2015:

CSMR 2014:	**17th European Conference on Software Maintenance and Reengineering** February 3-7, 2014, Antwerp, Belgium see: http://csmr.eu/ ***(not in 2015)***
ISEC 2015:	**8th India Software Engineering Conference** February 18 - 20, 2015, Bangalore, India see: http://isoft.acm.org/isec2015/

March 2015:

UKSMA 2015:	**Workshop on Defect Measurement and Analysis** March 4 , 2015, London, UK see http://uksma.co.uk/workshops.asp
ICSQ 2015:	**International Conference on Software Quality** March 9 - 11, 2015, Long Beach, California, USA see: http://www.asq-icsq.org/
REFSQ 2015:	**21th International Working Conference on Requirements Engineering: Foundation for Software Quality** March 23-26, 2014, Essen, Germany see: http://refsq.org/2015/-

April 2015:

FASE 2015:	**18th International Conference on Fundamental Approaches to Software Engineering** April 11-18, 2015, London, UK see: http://www.etaps.org/index.php/2015/fase
ISMA 2015:	**10th ISMA Conference of the IFPUG** April 30, 2015, Charlotte, North Caroline, USA see: http://www.ifpug.org/
ASWEC 2014:	**23nd Australian Software Engineering Conferences** April 7 - 10, 2014, Sydney, Australia see: http://www.aswec2014.org/ ***(not in 2015)***
ICST 2015:	**8th International Conference on Software Testing, Verification & Validation** April 13 - 17, 2015, Graz, Austria see: http://icst2015.ist.tu-graz.ac.at
ASQT 2015:	**Arbeitskonferenz Softwarequalität und Test** April 16 - 17., 2015, Graz, Austria see: http://www.asqt.org/
SOFTENG 2015	**First International Conference on Advances and Trends in Software Engineering** April 19 - 24, 2015, Barcelona, Spain see: http://www.iaria.org/conferences2015/SOFTENG15.html
CIbSE 2015:	**18th Iberoamerican Conference on Software Engineering** April 22-24, 2015, Lima, Peru see: https://sites.google.com/a/spc.org.pe/cibse2015/
CSEE&T 2014:	**26th Conference on Software Engineering Education and Training** April 23-25, 2014, Klagenfurt, Austria see: http://conferences.computer.org/cseet/2014/ ***(not in 2015)***
EASE 2015:	**19th International Conference on Empirical Assessment in Software Engineering** April 27-29, 2015, Nanjing, China see: http://emse.nju.edu.cn/ease2015/
iqnite 2015:	**Software Quality Conference** April 28 30, 2015, Düsseldorf, Germany see: http://www.iqnite-conferences.com/de/index.aspx
ENASE 2015:	**10th International Conference on Evaluation of Novel Approaches to Software Engineering** April 29 - 30, 2015, Barcelona, Spain see: http://www.enase.org/

May 2015:

WICSA 2015:	**12th Working IEEE/IFIP Conference on Software Architecture** May 4 - 8, 2015, Montreal, Canada see: http://wicsa2015.org/index.html
STAREAST 2015:	**Software Testing Analysis & Review Conference** May 3-8, 2014, Orlando, FL, USA see: http://stareast.techwell.com/
QoSA 2015:	**11th International ACM Sigsoft Conference on the Quality of Software Architectures** May 4 - 8, 2015, Montreal, Canada see: http://qosa.ipd.kit.edu/qosa_2015/
EMEA 2015:	**PMI Global Congress 2015 - EMEA** May 11-13, 2015,London, UK see: http://www.pmi.org/Learning/professional-development/Congress-PMI-Global-Congresses/EMEA-2015.aspx
SERA 2015:	**13th ACIS Conference on Software Engineering** May 13 - 15, 2015, Hammamet, Tunesia see: http://sera2015.redcad.org/
SAM 2015	**Workshop on Software Architecture and Metrics** May 16, 2015, Florence, Italy see: http://www.sei.cmu.edu/community/sam2015/
OSS 2015:	**11th International Conference on Open Source Systems** May 16 - 17, 2015, Florence, Italy see: http://www.oss2015.org/
ICSE 2015:	**37th International Conference on Software Engineering** May 16- 24, 2015, Florence, Italy see: http://2015.icse-conferences.org/
MSR 2015:	**11th Working Conference on Mining Software Repositories** May 16 - 17, 2014, Florence, Italy see: http://2015.msrconf.org/
ICPC 2015:	**22th International Conference on Program Comprehension** May 18 - 19, 2015, Florence, Italy see: http://www.program-comprehension.org/
XP 2015:	**16th International Conference on Agile Software Development** May 25-29, 2015, Helsinki, Finland see: http://www.xp2015.org/

June 2015:

EJC 2015:	**25th European Japanese Conference on Information Modeling and Knowledge Bases** June 9 - 12, 2015, Maribor, Slovenia see: http://ejc2015.um.si/
ICWE 2015:	**International Conference on Web Engineering** June 23 - 26, 2015, Rotterdam, Netherlands see: http://icwe2015.webengineering.org/
SPICE 2015:	**15th International SPICE Conference** June 16 - 17, 2015, Gothenburg, Sweden see: http://www.spiceconference.com/
SQ 2015	**Sixth International Symposium on Software Quality** June 22 - 25, 2015, Banff, Canada see: http://sq.covenantuniversity.edu.ng/

July 2015:

UKPEW 2014:	**24th Annual United Kingdom Workshop on Performance Engineering** July 4 - 5, 2014, Edinburgh, UK see: http://ukpew.lboro.ac.uk/ ***(not in 2015)***
VDA Automotive SYS Conference 2015:	**Quality Management for Automotive Software-based Systems and Functionality** July 15 - 17, 2015, Potsdam, Germany see: http://vda-qmc.de/software-prozesse/vda-automotive-sys/
ICSOFT 2015:	**10th International Conference on Software and Data Technologies** July 20 - 22, 2015, Colmar, Alsace, France see: http://www.icsoft.org/
SERP 2015	**13th Internatinal Conference on Software Engineering Research and Practice** July 27 - 30, 2015, Las Vegas, Nevada, USA see: http://www.world-academy-of-science.org/worldcomp15/ws/conferences/serp15
ICGSE 2015:	**10th International Conference on Global Software Engineering** July 13 - 16, 2015, Ciudad Real, Spain see: http://www.icgse.org/

August 2015:

AGILE 2015:	**International Conference on Agile** August 3 - 7, 2015, Washington D. C., USA see: http://agile2015.agilealliance.org/
RE 2015:	**23th IEEE International Requirement Engineering Conference** August 24-28, 2015, Ottawa, Canada see: http://re15.org/
Euromicro DSD/ SEAA 2015:	**41th Software Engineering & Advanced Application Conference** August 26 - 28, 2015, Funchal, Madeira, Portugal see: http://esd.scienze.univr.it/dsd-seaa-2015/

September 2015:

QEST 2015:	**12th International Conference on Quantitative Evaluation of Systems** September 1 - 3, 2015, Madrid, Spain see: http://www.qest.org/qest2015/
EuroSPI 2015:	**22th European Systems & Software Process Improvement and Innovation Conference,** September 20 - October 2, 2015, Ankara, Turkey see: http://www.eurospi.net/

October 2015:

IWSM-MENSURA 2015:	**Common International Conference on Software Measurement** October 5 - 7, 2015, Cracow, Poland see: http://www.iwsm-mensura.org/2015/cfp
ESEM 2015:	**9th International Symposium on Empirical Software Engineering & Measurement** October 22 - 23, 2015, Beijing, China see: http://eseiw.iscas.ac.cn/eseiw2015/esem/

November 2015:

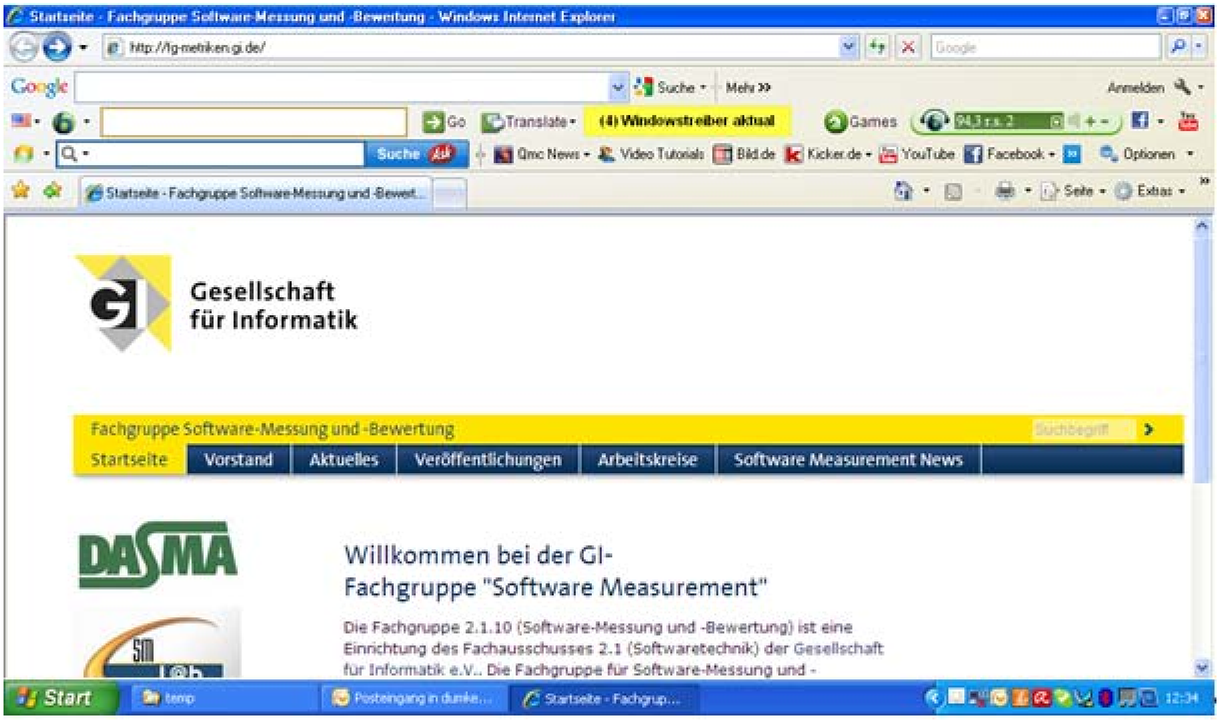
ISSRE 2015	**26th International IEEE Symposium on Software Reliability Engineering** November 2 - 5, 2015, Gaithersburg, USA see: http://issre.net/Invitation
BSOA/BCloud 2015:	**10. Workshop Bewertungsaspekte service-orientierte und Cloud-Architekturen** November , 2015, see: http://www-ivs.cs.uni-magdeburg.de/~gi-bsoa/
MetriKon 2015:	**International Conference on Software Measurement** November 5-6, 2015, Cologne, Germany see: http://www.metrikon.de/
ICSEA 2015	**10th International Conference on Software Engineering Advances** November 15 - 20, 2015, Barcelona, Spain see: http://www.iaria.org/conferences2015/ICSEA15.html

December 2015:

PROFES 2015:	**16th International Conference on Product Focused Software Process Improvement** December 2 - 4, 2015, Bolzano, Italy see: http://profes2015.inf.unibz.it/
ICSEFM 2015	**XIII International Conference on Software Engineering and Formal Methods** December 13 - 14, 2015, Melbourne, Australia see: https://www.waset.org/conference/2015/12/melbourne/ICSEFM

see also: Conferences Link of **Luigi Buglione (**http://www.semq.eu/leng/eveprospi.htm**)**

See the GI-Web site **http://fg-metriken.gi.de/** for the digital contents of the Software Measurement News:



Gesellschaft für Informatik

Fachgruppe Software-Messung und -Bewertung

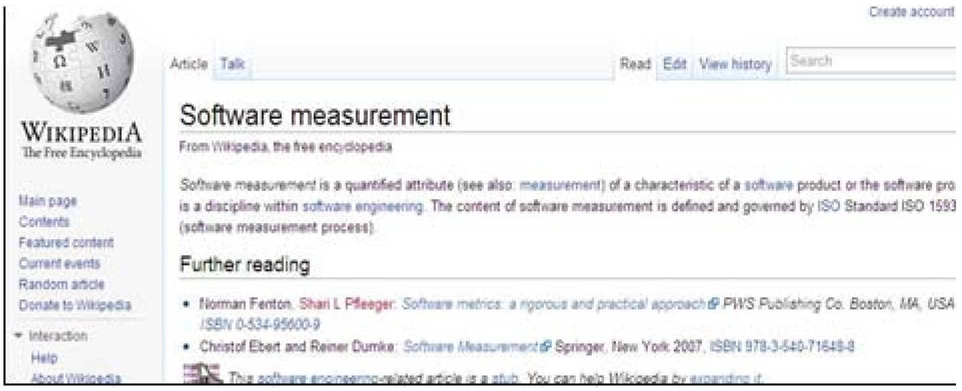
Startseite | Vorstand | Aktuelles | Veröffentlichungen | Arbeitskreise | Software Measurement News

DASMA

Willkommen bei der GI-Fachgruppe "Software Measurement"

Die Fachgruppe 2.1.10 (Software-Messung und -Bewertung) ist eine Einrichtung des Fachausschusses 2.1 (Softwaretechnik) der Gesellschaft für Informatik e.V.. Die Fachgruppe für Software-Messung und -

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



Software measurement

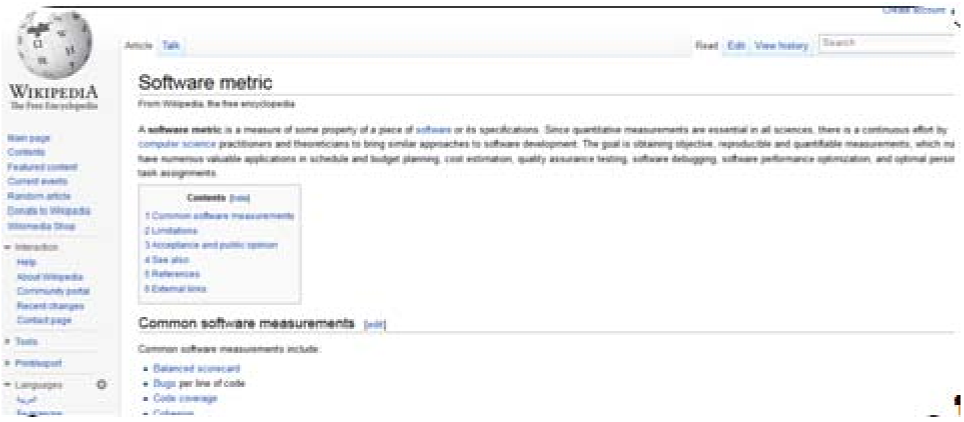
From Wikipedia, the free encyclopedia

Software measurement is a quantified attribute (see also: measurement) of a characteristic of a software product or the software pro
is a discipline within software engineering. The content of software measurement is defined and governed by ISO Standard ISO 1593
(software measurement process).

Further reading

- Norman Fenton, Shari L Pfleeger: Software metrics: a rigorous and practical approach PWS Publishing Co. Boston, MA, USA ISBN 0-534-95600-9
- Christof Ebert and Reiner Dumke: Software Measurement Springer, New York 2007, ISBN 978-3-540-71648-8

This software engineering-related article is a stub. You can help Wikipedia by expanding it.



Software metric

From Wikipedia, the free encyclopedia

A **software metric** is a measure of some property of a piece of software or its specifications. Since quantitative measurements are essential in all sciences, there is a continuous effort by computer science practitioners and theoreticians to bring similar approaches to software development. The goal is obtaining objective, reproducible and quantifiable measurements, which ma
have numerous valuable applications in schedule and budget planning, cost estimation, quality assurance testing, software debugging, software performance optimization, and optimal perso
task assignments.

Contents [hide]
1 Common software measurements
2 Limitations
3 Acceptance and public opinion
4 See also
5 References
6 External links

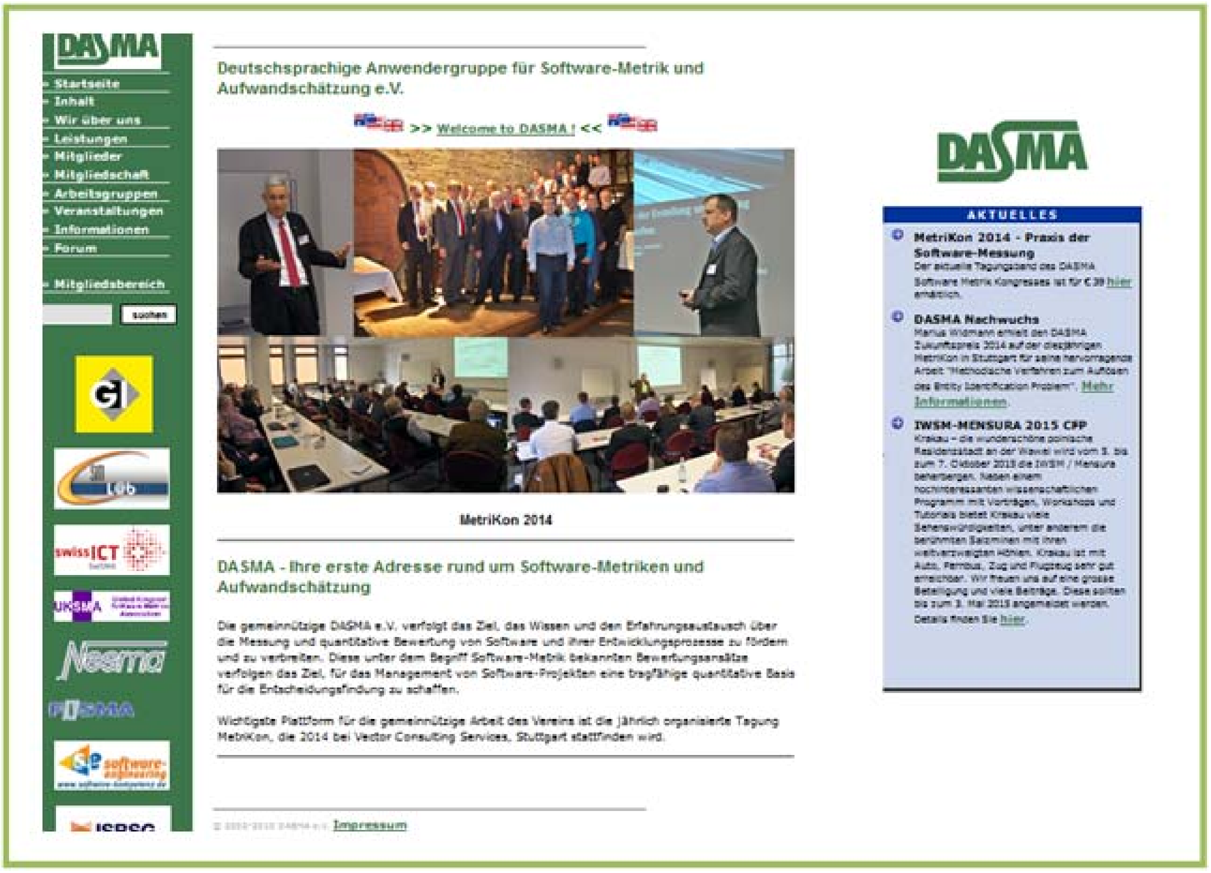
Common software measurements [edit]

Common software measurements include:

- Balanced scorecard
- Bugs per line of code
- Code coverage
- Cohesion

See our further software measurement and related communities:

www.dasma.org:



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

>> Welcome to DASMA ! <<

MetriKon 2014

DASMA - Ihre erste Adresse rund um Software-Metriken und Aufwandschätzung

Die gemeinnützige DASMA e.V. verfolgt das Ziel, das Wissen und den Erfahrungsaustausch über die Messung und quantitative Bewertung von Software und ihrer Entwicklungsprozesse zu fördern und zu verbreiten. Diese unter dem Begriff Software-Metrik bekannten Bewertungsansätze verfolgen das Ziel, für das Management von Software-Projekten eine tragfähige quantitative Basis für die Entscheidungsfindung zu schaffen.

Wichtigste Plattform für die gemeinnützige Arbeit des Vereins ist die jährlich organisierte Tagung MetriKon, die 2014 bei Vector Consulting Services, Stuttgart stattfinden wird.

© 2002-2015 DASMA e.V. Impressum

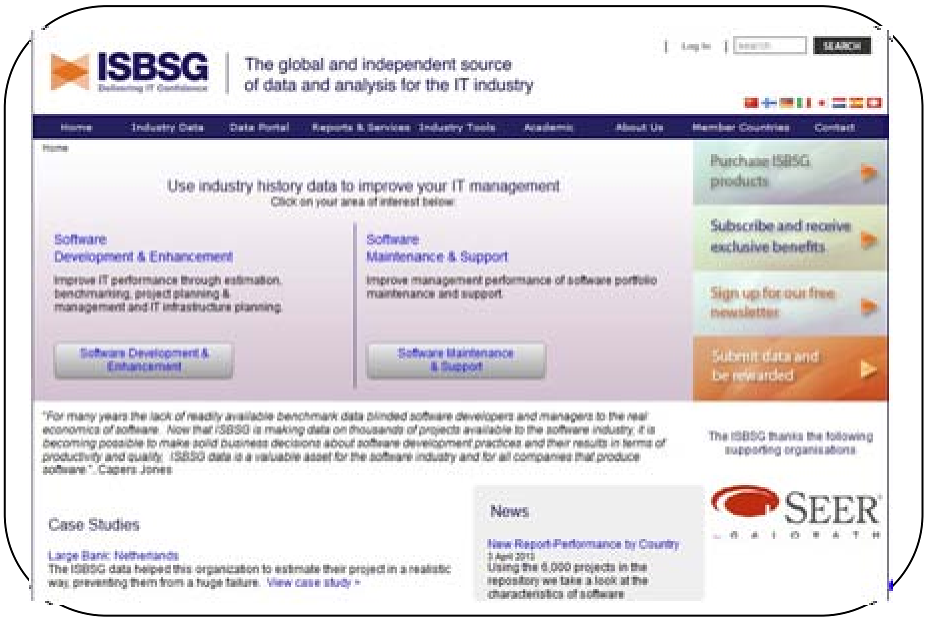
AKTUELLES

MetriKon 2014 - Praxis der Software-Messung
Der aktuelle Tagungsband des DASMA Software Metrik Kongresses ist für € 39 hier erhältlich.

DASMA Nachwuchs
Marius Widmann erhielt den DASMA Zukunftspreis 2014 auf der diesjährigen MetriKon in Stuttgart für seine hervorragende Arbeit "Methodische Verfahren zum Auflösen des Entity Identification Problem". Mehr Informationen.

IWSM-MENSURA 2015 CFP
Krakau – die wunderschöne polnische Residenzstadt an der Wawel wird vom 5. bis zum 7. Oktober 2015 die IWSM / Mensura beherbergen. Neben einem hochinteressanten wissenschaftlichen Programm mit Vorträgen, Workshops und Tutorials bietet Krakau viele Sehenswürdigkeiten, unter anderem die berühmten Salzminen mit ihren weitverzweigten Höhlen. Krakau ist mit Auto, Fernbus, Zug und Flugzeug sehr gut erreichbar. Wir freuen uns auf eine grosse Beteiligung und viele Beiträge. Diese sollten bis zum 3. Mai 2015 angemeldet werden. Details finden Sie hier.

www.isbsg.org:



ISBSG Delivering IT Confidence

The global and independent source of data and analysis for the IT industry

Log In
SEARCH
Home
Industry Data
Data Portal
Reports & Services
Industry Tools
Academic
About Us
Member Countries
Contact
Purchase ISBSG products
Subscribe and receive exclusive benefits
Sign up for our free newsletter
Submit data and be rewarded
Software Development & Enhancement
Software Maintenance & Support

Home

Use industry history data to improve your IT management
Click on your area of interest below

Software Development & Enhancement
Improve IT performance through estimation, benchmarking, project planning & management and IT infrastructure planning.

Software Maintenance & Support
Improve management performance of software portfolio maintenance and support.

"For many years the lack of readily available benchmark data blinded software developers and managers to the real economics of software. Now that ISBSG is making data on thousands of projects available to the software industry, it is becoming possible to make solid business decisions about software development practices and their results in terms of productivity and quality. ISBSG data is a valuable asset for the software industry and for all companies that produce software". Capers Jones

The ISBSG thanks the following supporting organisations

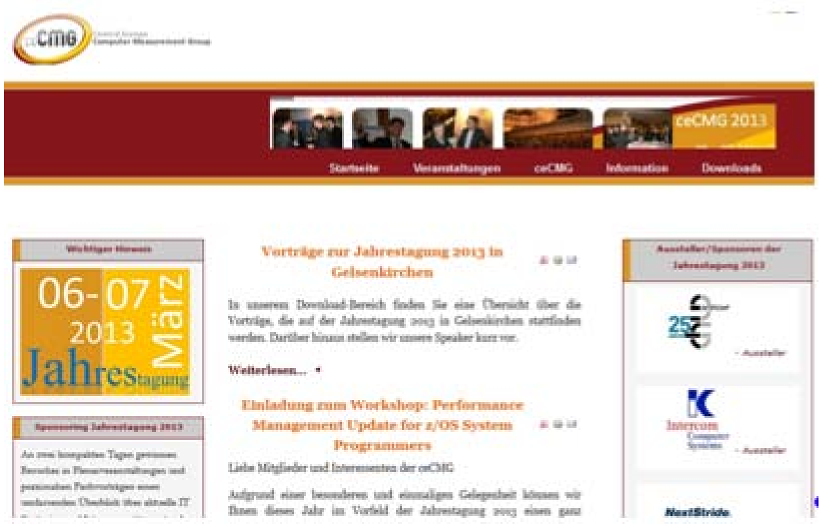
Case Studies

Large Bank: Netherlands
The ISBSG data helped this organization to estimate their project in a realistic way, preventing them from a huge failure. View case study >

News

New Report-Performance by Country
3 April 2013
Using the 6,000 projects in the repository we take a look at the characteristics of software

www.cecmg.de:


ceCMG 2013
Startseite
Veranstaltungen
ceCMG
Information
Downloads
Wichtiger Hinweis
06-07 März 2013 Jahrestagung
Sponsoring Jahrestagung 2013
Aussteller/Sponsoren der Jahrestagung 2013
- Aussteller
Intercom Computer Systems
- Aussteller
NextStride

Vorträge zur Jahrestagung 2013 in Gelsenkirchen

In unseren Download-Bereich finden Sie eine Übersicht über die Vorträge, die auf der Jahrestagung 2013 in Gelsenkirchen stattfinden werden. Darüber hinaus stellen wir unsere Speaker kurz vor.

Weiterlesen...

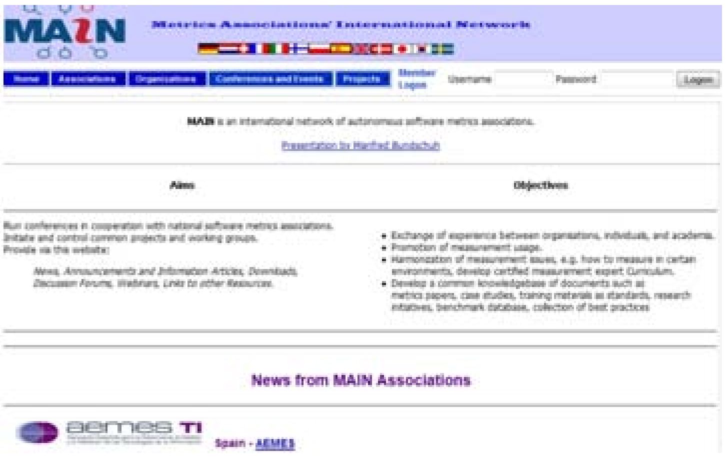
Einladung zum Workshop: Performance Management Update for z/OS System Programmers

Liebe Mitglieder und Interessenten der ceCMG

Aufgrund einer besonderen und einmaligen Gelegenheit können wir Ihnen dieses Jahr im Vorfeld der Jahrestagung 2013 einen ganz

An zwei kompakten Tagen gewinnen Besucher in Plenarveranstaltungen und praxisnahen Fachvorträgen einen umfassenden Überblick über aktuelle IT

www.mai-net.org:



MAIN is an international network of autonomous software metrics associations.

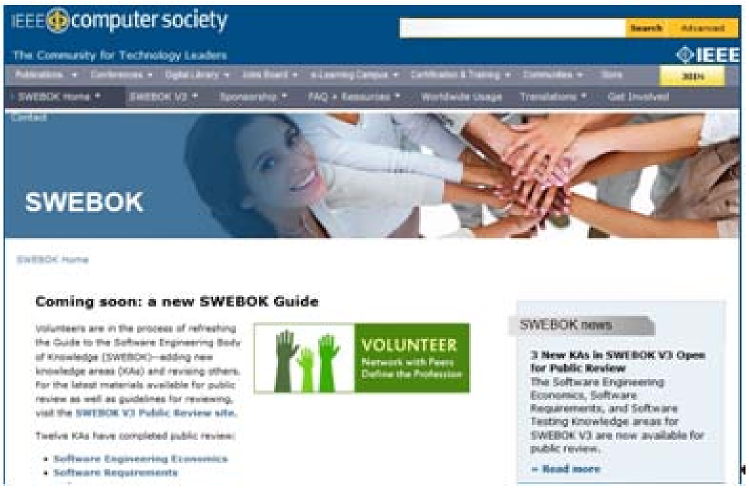
Presentation by Manfred Bundschuh

Aims	Objectives
Run conferences in cooperation with national software metrics associations. Initiate and control common projects and working groups. Provide via this website: *News, Announcements and Information Articles, Downloads, Discussion Forums, Webinars, Links to other Resources.*	• Exchange of experience between organizations, individuals, and academia. • Promotion of measurement usage. • Harmonization of measurement issues, e.g. how to measure in certain environments, develop certified measurement expert Curriculum. • Develop a common knowledgebase of documents such as metrics papers, case studies, training materials as standards, research initiatives, benchmark database, collection of best practices

News from MAIN Associations

Spain - AEMES

www.swebok.org:



SWEBOK Home

Coming soon: a new SWEBOK Guide

Volunteers are in the process of refreshing the Guide to the Software Engineering Body of Knowledge (SWEBOK)—adding new knowledge areas (KAs) and revising others. For the latest materials available for public review as well as guidelines for reviewing, visit the SWEBOK V3 Public Review site.

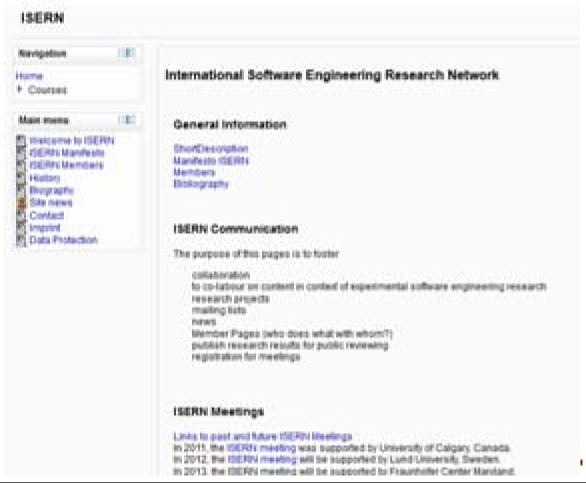
Twelve KAs have completed public review:

- Software Engineering Economics
- Software Requirements

3 New KAs in SWEBOK V3 Open for Public Review
The Software Engineering Economics, Software Requirements, and Software Testing Knowledge areas for SWEBOK V3 are now available for public review.

» Read more

isern.iese.de:



ISERN

Navigation
Home
Courses

Main menu
Welcome to ISERN
ISERN Manifesto
ISERN Members
History
Biography
Site news
Contact
Imprint
Data Protection

International Software Engineering Research Network

General Information

ShortDescription
Manifesto ISERN
Members
Bibliography

ISERN Communication

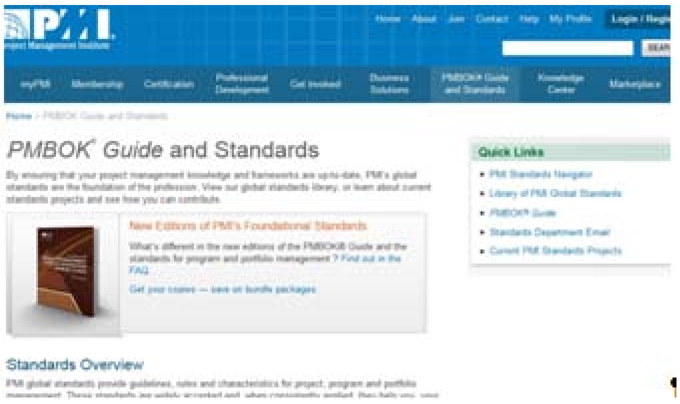
The purpose of this pages is to foster

collaboration
to co-labour on content in context of experimental software engineering research
research projects
mailing lists
news
Member Pages (who does what with whom?)
publish research results for public reviewing
registration for meetings

ISERN Meetings

Links to past and future ISERN Meetings
In 2011, the ISERN meeting was supported by University of Calgary, Canada.
In 2012, the ISERN meeting will be supported by Lund University, Sweden.
In 2013, the ISERN meeting will be supported by Fraunhofer Center Maryland.

www.pmbok.org:



Home About Join Contact Help My Profile Login / Register

myPMI Membership Certification Professional Development Get Involved Business Solutions PMBOK® Guide and Standards Knowledge Center Marketplace

Home > PMBOK Guide and Standards

PMBOK® Guide and Standards

By ensuring that your project management knowledge and frameworks are up-to-date, PMI's global standards are the foundation of the profession. View our global standards library, or learn about current standards projects and see how you can contribute.

New Editions of PMI's Foundational Standards

What's different in the new editions of the PMBOK® Guide and the standards for program and portfolio management? Find out in the FAQ.

Get your copies — save on bundle packages

Quick Links
- PMI Standards Navigator
- Library of PMI Global Standards
- PMBOK® Guide
- Standards Department Email
- Current PMI Standards Projects

Standards Overview

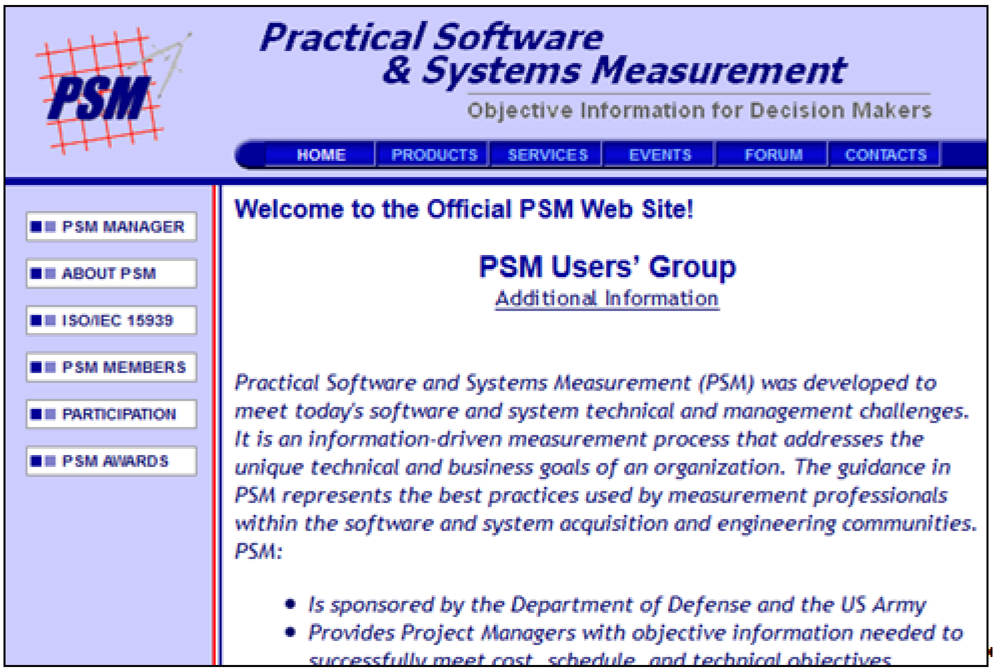
PMI global standards provide guidelines, rules and characteristics for project, program and portfolio management. These standards are widely accepted and, when consistently applied, they help you, your

www.smlab.de:

Virtuelles Softwaremesslabor der Uni Magdeburg
Otto-von-Guericke-University of Magdeburg
Software Measurement Laboratory SML@b
Current Events
IWSM/Mensura Conference
BSOA Workshop
MetriKon Conference
DASMA
cecmg
ISBSG
General Informations
Conferences
Measurement Approaches & Tools
Software Measurement News
Books
PhD Thesis
Master/Diploma/Bachelor Thesis
Publications
Measurement Education
Keynotes & Research Topics
CuBIT
NAIST

Metrics in the World-Wide Web

www.psmsc.com/:



Practical Software & Systems Measurement

Objective Information for Decision Makers

Welcome to the Official PSM Web Site!

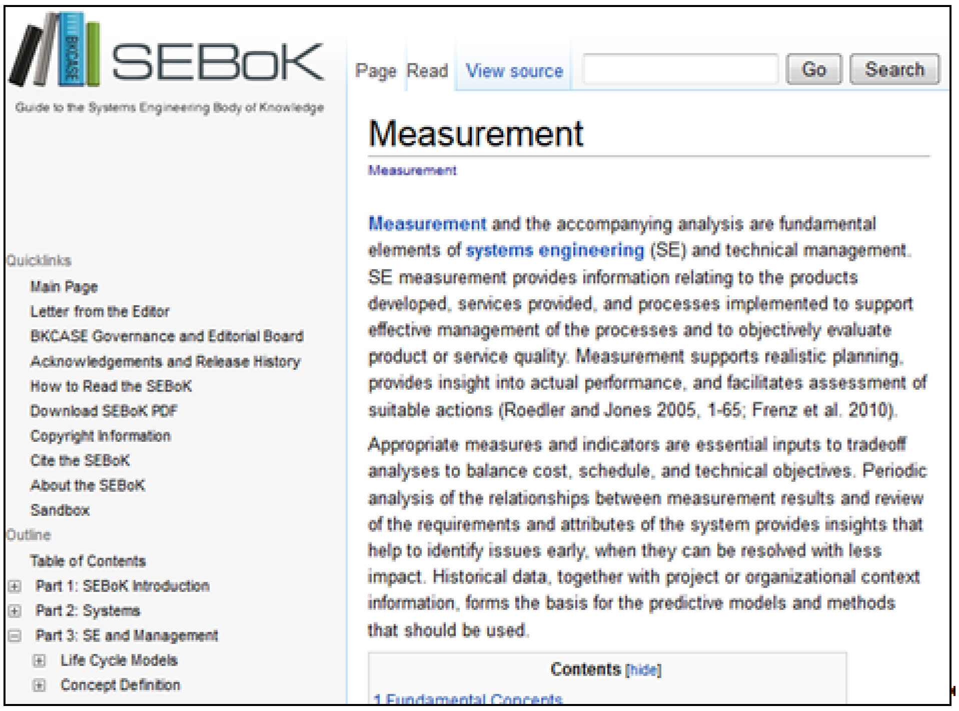
PSM Users' Group

Additional Information

Practical Software and Systems Measurement (PSM) was developed to meet today's software and system technical and management challenges. It is an information-driven measurement process that addresses the unique technical and business goals of an organization. The guidance in PSM represents the best practices used by measurement professionals within the software and system acquisition and engineering communities. PSM:

- *Is sponsored by the Department of Defense and the US Army*
- *Provides Project Managers with objective information needed to successfully meet cost, schedule, and technical objectives*

sebokwiki.org/wiki/Measurement:



Measurement

Measurement

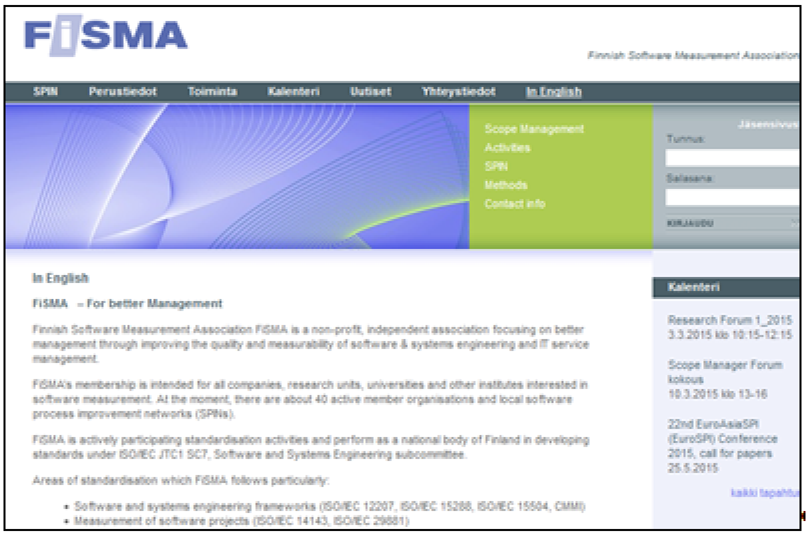
Measurement and the accompanying analysis are fundamental elements of systems engineering (SE) and technical management. SE measurement provides information relating to the products developed, services provided, and processes implemented to support effective management of the processes and to objectively evaluate product or service quality. Measurement supports realistic planning, provides insight into actual performance, and facilitates assessment of suitable actions (Roedler and Jones 2005, 1-65; Frenz et al. 2010).

Appropriate measures and indicators are essential inputs to tradeoff analyses to balance cost, schedule, and technical objectives. Periodic analysis of the relationships between measurement results and review of the requirements and attributes of the system provides insights that help to identify issues early, when they can be resolved with less impact. Historical data, together with project or organizational context information, forms the basis for the predictive models and methods that should be used.

Contents [hide]

1 Fundamental Concepts

www.fisma.fi/in-english/:



In English

FiSMA – For better Management

Finnish Software Measurement Association FiSMA is a non-profit, independent association focusing on better management through improving the quality and measurability of software & systems engineering and IT service management.

FiSMA's membership is intended for all companies, research units, universities and other institutes interested in software measurement. At the moment, there are about 40 active member organisations and local software process improvement networks (SPINs).

FiSMA is actively participating standardisation activities and perform as a national body of Finland in developing standards under ISO/IEC JTC1 SC7, Software and Systems Engineering subcommittee.

Areas of standardisation which FiSMA follows particularly:

- Software and systems engineering frameworks (ISO/IEC 12207, ISO/IEC 15288, ISO/IEC 15504, CMMI)
- Measurement of software projects (ISO/IEC 14143, ISO/IEC 29881)

Kalenteri

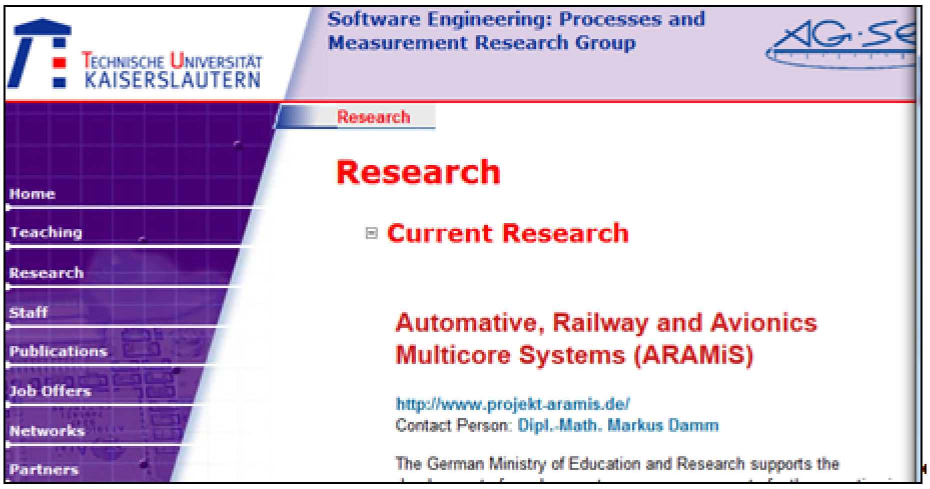
Research Forum 1_2015
3.3.2015 klo 10:15-12:15

Scope Manager Forum kokous
10.3.2015 klo 13-16

22nd EuroAsiaSPI (EuroSPI) Conference 2015, call for papers 25.5.2015

kaikki tapahtu

wwwagse.informatik.uni-kl.de/research/:



Research

⊟ **Current Research**

Automative, Railway and Avionics Multicore Systems (ARAMiS)

http://www.projekt-aramis.de/
Contact Person: Dipl.-Math. Markus Damm

The German Ministry of Education and Research supports the

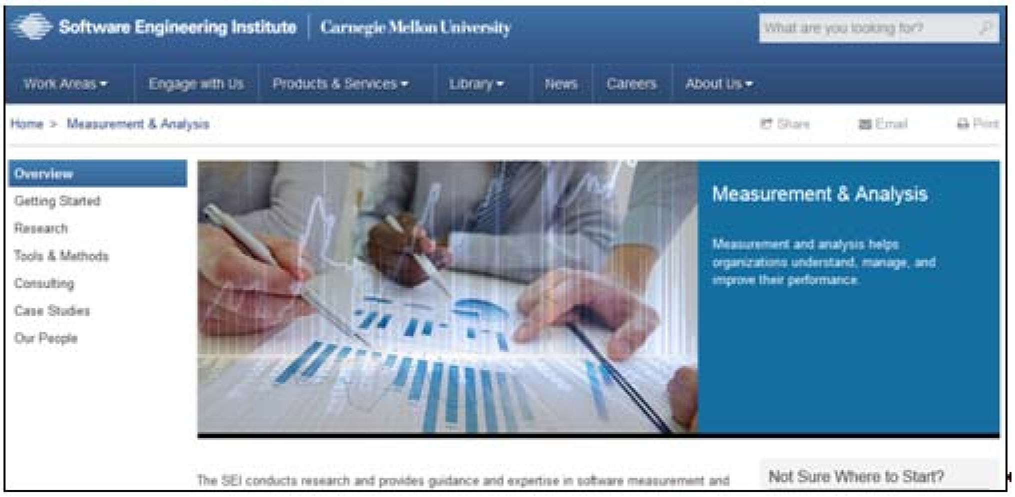
http://nesma.org/:

nesma
MEN
TIME IS MONEY
Estimating
It is not about the size. Estimating is the translation of size to the things that really matter: time and money.
SHOW ME THE MONEY
GIVE ME SOME TIME

Your starting point for software measurement and software

Metrics in the World-Wide Web

www.sei.cmu.edu/measurement/:

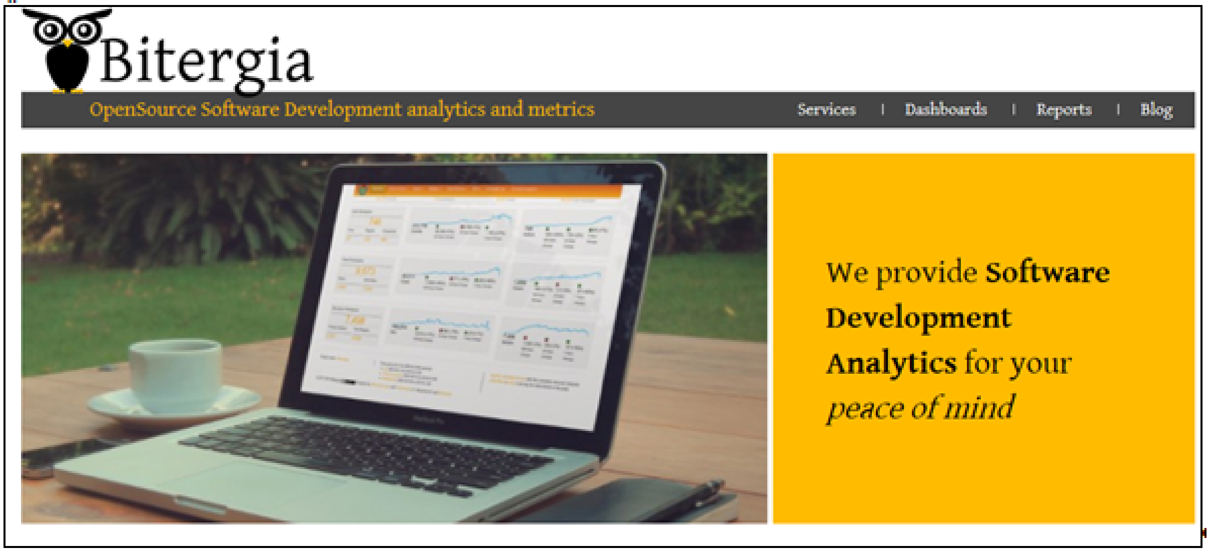


Measurement & Analysis

Measurement and analysis helps organizations understand, manage, and improve their performance.

The SEI conducts research and provides guidance and expertise in software measurement and

bitergia.com/:



We provide **Software Development Analytics** for your *peace of mind*

SOFTWARE MEASUREMENT NEWS

VOLUME 20 2015 NUMBER 1

CONTENTS

ISSN 1867-9196