METRICS NEWS

VOLUME 9 *2004* *NUMBER 1*

CONTENTS

ISSN 1431-8008

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

Editors:

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

MANFRED BUNDSCHUH
Chair of the DASMA
Sander Höhe 5, 51465 Bergisch Gladbach, Germany
***Tel.:** +49-2202-35719*
Bundschuhm@acm.org
http://www.dasma.org

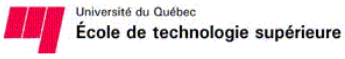
REINER DUMKE
Professor on Software Engineering
University of Magdeburg, FIN/IVS
Postfach 4120, D-39016 Magdeburg, Germany
***Tel.:** +49-391-67-18664, **Fax:** +49-391-67-12810*
dumke@ivs.cs.uni-magdeburg.de

CHRISTOF EBERT
Dr.-Ing. in Computer Science and Director SW Coordination
Alcatel HQ
54, Rue La Boetie, F-75008 Paris, France
***Tel.:** +33-675-091999, **Fax:** +33-1-4076-1475*
christofebert@ieee.org

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

Editorial Office: Otto-von-Guericke-University of Magdeburg, FIN/IVS, Postfach 4120, 39016 Magdeburg, Germany
Technical Editor: DI Mathias Lother
The journal is published in one volume per year consisting of two numbers.

CALL FOR PARTICIPATION



IWSM2004

14th International Workshop
on Software Measurement

MetriKon2004

DASMA Metrik Kongress

of the **DASMA**- Deutschsprachige Anwendergruppe für Software-Metrik und Aufwandschätzung
GI FG 2.1.10 - German Interest Group on Software Metrics and the
CIM - Canadian Interest Group on Metrics
COSMIC – Common Software Measurement International Consortium
MAIN – International Network of Metrics Associations
In cooperation with:
University of Magdeburg, Germany, École de Technologie Supérieure - Université du Québec, Canada, and T-Systems, Germany

November 2-5, 2004, Berlin, Königs Wusterhausen

http://iwsm2004.cs.uni-magdeburg.de, http://www.dasma.org

THEME & SCOPE

Software measurement and metrics application are some of the key technologies to control or to manage the software development process. Measurement is also the foundation of both sciences and engineering, and much more research in software is needed to ensure that software engineering be recognized as a true engineering discipline in order to keep IT companies successful in the marketplace.

TOPICS OF INTEREST

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

Software metrics foundations
Practical measurement application
Measurement processes and resources
Empirical case studies
Measurement acceptance
Software estimation
Measurement services
Functional size measurement
Software process improvement

Metrics validation
Measurement data bases
Web metrics

Measurement tool support and infrastructures
Measurement experience and guidance
Theory of measurement
Measurement paradigms
Enterprise embedded solutions

PROGRAM COMMITTEE

ADVANCED PROGRAM

DASMA MetriKon 2004

2. November 2004

09.00 **Tutorials**

Walter Tichy, Uni Karlsruhe

Claus Lewerentz, Uni Cottbus
„Metrikbasierte Qualitätsanalysen I“

13:00 *Lunch Break - Networking*

14:00 **Tutorials or Community Sessions**

Ralf Kalmar, Fraunhofer IESE, Kaiserslautern

Claus Lewerentz, Uni Cottbus
„Metrikbasierte Qualitätsanalysen II“

17:30 *Break*

18:30 **Exhibitors’ Reception**

DASMA Vorstandssitzung / DASMA Board Meeting

3. November 2004

08:00 Conference Registration **Wintergarten**

09:00 Welcome and Introduction **Room 155**

09:30 **Keynote I** **Room 155**

Pekka Forselius:
Making a move from function point counting to better project management

---

10:30 *Coffee Break - Networking - Exhibition*

11:00 **Session A1** **Room 155**

Eberhard Rudolph:

Measuring the size of application software overheads

M. Lother, R. Braungarten, M. Kunz, R. Dumke:
The Functional Size eMeasurement Portal (FSeMP) - A Web-based Approach for Effort Estimation, Benchmarking and eLearning

Luca Santillo:
Software complexity evaluation based on functional size components

11:00 **Session B1** **Room 156**

José A. Cruz-Lemus, Marcela Genero, Mario Piattini:
Validating Metrics for UML Statechart - Diagrams Through a Family of Experiments

Edgardo Palza, Alain Abran, Christofer Fuhrman:
V&V Measurements Management Issues in Safety-Critical Software
George Wilkie, M.P. Ware, B.A. Kitchenham, T.J. Harmer:
Evaluating the Sensitivity of Coupling Metrics to Evolving Software Systems

---

12:30 *Lunch Break - Networking - Exhibition*

12.30 – 13.00 Industrial session 1
14:00 – 14.30 Industrial session 2

14:30 **Session A2 - IWSM** **Room 155**

Alain Abran, Adel Khelifi:
A System of References for Software Measurements with ISO 19761 (COSMIC-FFP)

F.W.Vogelezang:
Implementing COSMIC FFP as a replacement for FPA

J. Cuadrado-Gallego, J. Dolado, D. Rodríguez, M. Sicilia:
The Second Level Input Variables for Software Cost Estimation Models

14:30 **Session B2 - MetriKon** **Room 156**

Lutz Winkler, Frank Schmeißner:
ERP-Standard-SW-Anbieter im magischen Dreieck von Arbeitsweise, Kosteneffizienz und Produktqualität

A. Schmietendorf, D. Reitz, J. Lezius, E. Dimitrov, T. Walter:
Aufwandsschätzung neuer Integrationsanforderungen im Rahmen einer bereits etablierten Integrationslösung

---

15:30 *Coffee Break - Networking - Exhibition*

16:30 **Session A3 - IWSM** **Room 155**

Tom Koppenberg:
Estimating maintenance projects using COSMIC FFP

De Tran-Cao, Ghislain Lévesque, Jean-Guy Meunier:
A Field Study of Software Functional Complexity Measurement

Alain Abran, Blanca Gil:
Statistical analysis of Function Point profiles

16:30 **Session B3 - MetriKon** **Room 156**

Johannes Drexler, Francesca Saglietti:
Eine einheitliche Kohäsionsmetrik für Methoden, Klassen und Komponenten

Richter/Simon:
Mit Code-Metriken Wartungskosten senken: Controlling technischer Qualität

---

18:00 *Break*

19:00 Community Sessions - Networking

DASMA Mitgliederversammlung (DASMA annual general meeting)	Room 156
GI FG 2.1.10 Mitgliederversammlung GI FG 2.1.10 annual general meeting	Room 155
COSMIC IAC - Meeting	Room tba

4. November 2004

08:30 **Session A4** **Room 155**

Alain A. April, Alain Abran, Reiner Dumke:
Software Maintenance Productivity measurement: how to Assess the readiness of your organization

R. Dumke, M. Lother, U. Schäfer, C. Wille:
Web Tomography - Towards an e-Measurement and Controlling

Alain Abran, Luigi Buglione, Asma Sellami:
Software Measurement Body of Knowledge - Initial Validation using Vincenti's Classification of Engineering Knowledge

08:30 **Session B4** **Room 156**

Christof Ebert:
Portfolio-Management for Software Projects

Thomas Fehlmann:

Metrics for Cooperative Development Processes

10:00 *Coffee Break - Networking - Exhibition*

10:30 **Session A5 - IWSM** **Room 155**

J.-M. Desharnais, A. Abran, J. Vilz, F. Gruselin, N. Habra:
Verification and validation of a knowledge-based system

Cornelius Wille, Reiner R. Dumke, Nick Brehmer:
Evaluation of Agent Academy: Measurement Intentions

P. Bourque, S. Wolff, R. Dupuis, A. Sellami, A. Abran:
Lack of Consensus on Measurement in Software Engineering: Investigation of Related Issues

10:30 **Session B5 - MetriKon** **Room 156**

Marek Leszak:
The Versatility of Software Defect Prediction Models (or why it's so hard to replicate related Case Studies)

Hans-Georg Hopf:
Software Reliability - Grundlagen und Berechnung

12:00 *Lunch Break - Networking - Exhibition*

12.00 – 12.30 Industrial session 3
13:30 – 14.00 Industrial session 4

14:00 **Session A6 – IWSM** **Room 155**

Hamdan Msheik, Alain Abran, Hamid Mcheick:
Measuring Components Unused Members

Andreas Schmietendorf, Reiner Dumke:
A Measurement Service for Monitoring the Quality Behaviour of Web Services offered within the Internet

Alain Abran, Miguel Lopez, Naji Habra:
An Analysis of the Mc Cabe Cyclomatic Complexity Number

14:00 **Session B6 - MetriKon** **Room 156**

Bela Mutschler, Manfred Reichert:
Usability Metriken als Nachweis der Wirtschaftlichkeit von Verbesserungen der Mensch Maschine Schnittstelle

Jörg Robra:

Kreative Software-Messung - Kleiner Leitfaden statistischer Tricks

Rüdiger Liskowsky:
Bewertung der Gebrauchstauglichkeit mit Metriken

15:30 *Coffee Break - Networking - Exhibition*

16:00 **Session A7 - IWSM** **Room 155**

Arlan Lesterhuis, F.W.Vogelezang:
The COSMIC FFP Business Applications Guideline

Alain Abran, Olga Ormandjieva, Manar Abu Talib:
Functional Size and Information Theory-Based Functional Complexity Measures: Exploratory study of related concepts using COSMIC-FFP measurement method as a case study

Juan Carlos Granja-Alvarez:
Function Points Analysis Based on Requirement Specification, a Case Study

16:00 **Session B7 - MetriKon** **Room 156**

Robert Hürten:
Ergebnis einer internationalen Befragung zur Einführung und Nutzung der Software Metrik

Jürgen Bach, Björn Petersdorf:
Zwischen Wunsch und Wirklichkeit: Einführung von Kennzahlensystemen in die IT-Projekt- und Unternehmenssteuerung

Der Träger des DASMA Diplomarbeitenpreises 2004 stellt seine prämierte Arbeit vor.

17:30 *Break*

19:00 **Social Event**

IWSM / MetriKon 2004

5. November 2004

09:00 **Session A8 - Thema** **Room 155**

Olga Jaufman:
Reusage Knowledge on Process Flexibility for Developing Measurement Programs

Melanie Ruhe:

How do we measure process improvement?Examples from industry

Ton Dekkers:
IT Governance requires quantitative (project) management

09:00 **Session B8 - Thema** **Room 156**

Maya Daneva:
Patterns of Success or Failure in ERP Requirements Engineering: an Empirical Study

D. Natale, L. Santillo, I. Della Noce, S. Lombardi, G. Moretto:
Proposals for project collection and classification from the analysis of the ISBSG Benchmark

Roland Neumann, Stamatia Bibi:
Building Fault Prediction Models from Abstract Cognitive Complexity Metrics – Analysing and Interpreting Fault Related Influences

---

10:30 *Coffee Break - Networking - Exhibition*

11:00 **Keynote II**

Pam Morris:
Metrics based Project Governance

12:00 Closing Session

12:30 *Lunch Break - Networking – Exhibition*

Ankündigung des 6. Workshop „Performance Engineering in der System- und Softwareentwicklung“ (PE2005) im Frühjahr des Jahres 2005 an der Fachhochschule Köln

Der Performance Engineering Arbeitskreis (kurz PEAK) der GI-Fachgruppe 2.1.10 (Software-Messung und -Bewertung) beschäftigt sich mit dem Performance Engineering in der System- und Softwareentwicklung. Wenngleich die Auswertung und Verarbeitung der Ergebnisse des diesjährigen Workshops noch nicht abgeschlossen sind, gilt es bereits jetzt an den Workshop im kommenden Jahr zu denken, welcher an der Fachhochschule Köln - Fachbereich Informatik (Mai oder Juni) stattfinden wird. Im Folgenden finden sich einige Themen zu potentiellen Beiträgen des kommenden Workshops.

Performance Management und Performance Messungen:

- Performance Management von Softwareanwendungen
- Techniken der Performancemessung für Services
- Performanceadaptive Lösungen (Komponenten/Services/Agenten)

Techniken der Modellierung und verfügbare Werkzeuge:

- Modell driven Architecture (OMG MDA) und SPE
- Modellierung von Nutzerverhalten und QoS-Anforderungen
- XML-basierte Performance Spezifikationen

Industrielle Software Performance Engineering Prozesse

- Performance-orientierte Phasen der Softwareentwicklung
- Mehrwertpotentiale und Aufwände für das Performance Engineering
- Management von Performanceanforderungen

Unterschiede und Gemeinsamkeiten der System- und Softwareentwicklung

- Einsatz von Hardware vs. Entwicklung/Einsatz von Software
- Performanceeigenschaften beim System- und Architekturentwurf
- Standardisierung von Hard- und Softwarekomponenten

Wie in jedem Jahr wird sich der Workshop wieder als Diskussionsforum zu aktuellen Herausforderungen im Umfeld des Performance Engineering verstehen und auf aktuelle Trends eingehen. Dabei werden insbesondere die Inhalte internationaler Workshops (CMG-Conference, WOSP, UKPEW, …) im Umfeld des Performance Engineering beachtet. Weitere Hinweise zu den entsprechenden Terminen für einzureichende Beiträge finden sich auf der Webseite des Arbeitskreises.

Webseite des Arbeitskreises:

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

Ansprechpartner des Arbeitskreises:

andreas.schmietendorf@t-systems.com

Bericht zum 5. Workshop „Performance Engineering in der System- und Softwareentwicklung“ (PE 2004)

Andreas Schmietendorf (Sprecher des GI-PEAK)
andreas.schmietendorf@t-systems.com

Überblick und Motivation

Der im Jahr 2000 gegründete Arbeitskreis zum Performance Engineering (kurz GI-PEAK) führte am 14. Mai 2004 bereits zum 5. Mal seinen jährlich stattfindenden Workshop durch. Gastgeber in diesem Jahr war die Siemens AG in München. An dieser Stelle sei noch einmal Herrn Dr. Stefan Rugel für die hervorragende Organisation herzlich gedankt. Wie in den vorherigen Jahren konnte auf der Basis der eingereichten Beiträge wieder ein anspruchsvolles Workshop-Programm zusammengestellt werden. Abgerundet wurde es durch eine entsprechende Panel-Diskussion zu aktuellen Herausforderungen im Umfeld des Performance Engineerings. Für diejenigen Leser, die unseren Arbeitskreis noch nicht kennen, soll im Folgenden noch einmal die Arbeitsdefinition des GI-PEAK zum Performance Engineering aufgezeigt werden. Es handelt sich nicht um eine statische Festlegung, vielmehr erfolgt eine zyklische Korrektur bzw. Anpassung entsprechend der neuen Erkenntnisse bzw. aktuellen Herausforderungen.

> *"Performance Engineering versteht sich als Methode zur Berücksichtigung von zeit- und ressourcenbezogenen Qualitätszielen während der System- und Softwareentwicklung. Dabei sind sowohl wirtschaftliche als auch technische Randbedingungen zu berücksichtigen bzw. unter Performancegesichtspunkten zu determinieren."*

Im Unterschied zu den entsprechenden Performance Engineering Workshops in den USA (WOSP – Workshop on Software and Performance) und Großbritannien (UKPEW – United Kingdom Performance Engineering Workshop) konnten wir auch in diesem Jahr mit fast 70% der Teilnehmer wieder eine starke Beteiligung aus dem industriellen Umfeld feststellen.

Beiträge des Workshops

Die Beiträge des Workshops bildeten einen Canon zu Themen aus dem industriellen aber auch akademischen Umfeld, wobei der Schwerpunkt - wie bereits in den vergangenen Jahren - aus der Industrie kommt. So hatten wir in diesem Jahr Praxisbeiträge der Firmen Siemens, Lucent, Softlab und der T-Systems sowie theoretische Arbeiten von den Universitäten Magdeburg, Paderborn und Plovdiv (Bulgarien). Insgesamt beteiligten sich in diesem Jahr 20 Autoren an den durch das Programmkomitee akzeptierten Beiträgen; auch dies stellt in der Geschichte des Workshops einen kleinen Rekord dar. Im Folgenden findet sich die Übersicht zu den entsprechenden Vorträgen bzw. Postern.

- *Dieter Stoll*: Performance Engineering for large Embedded Systems in Practice
- *Ndombe Cacutalua*: Mehr Wert generieren unter Weiterverwendung des Bewährten: Integration einer COBOL-Anwendung in ein J2EE-Umfeld
- *Andreas Hennig, Rainer Wasgint, Lev Olkhovic, Boris Petrovic*: Instant Performance Prototyping of EJB/J2EE Applications – A car rental example
- *Reiner Dumke, Uwe Schäfer, Cornelius Wille, Fritz Zbrog*: Agentenbasierte Web-Technologiebewertung für das Performance Engineering

- *Andreas Schmietendorf, Daniel Reitz, Dimitry Rud*: Performancebetrachtungen im Umfeld webservice-basierter Integrationslösungen
- *Dietmar Weber, Antonius Erdmann*: Performance measurements and prognosis for large-scale multi-processor telecommunication systems
- *Hans Mauser, Christoph Wincheringer*: Performance Analysis of the IP Multimedia Subsystem for 3rd Generation Mobil Communication
- *Stanimir Stojanov*: MALINA – eine agenten-orientierte Entwicklungsumgebung
- *Henner Diederichs, Leena Suhl*: Rescueing Software Projects through Complexity Reduction - a System Theoretic Approach

Abbildung 1: Tagungsband PE2004

Alle Beiträge sind im Tagungsband [Schmietendorf/Dumke 2004] zum Workshop enthalten, der über den Sprecher des Arbeitskreises zu einem Unkostenbeitrag von 15,- Euro noch bezogen werden kann.

Ergebnisse der Panel-Diskussion

Die bereits auf dem letzten Workshop des GI-PEAK eingeführte Panel-Diskussion wurde auch in diesem Jahr wieder zu einem interessanten Diskussionsforum. Angeregt wurde die Diskussion von den folgenden Fragestellungen, die im Kontext des Workshops identifiziert wurden:

- Wer ist der Kunde für das Performance Engineering?
- Auswirkungen von SOA auf das Performance Engineering?

- Welche Themen sollte der GI-PEAK Workshop zukünftig aufgreifen?

Im Rahmen eines lockeren, aber dennoch sehr lebhaft geführten Gesprächs zwischen den Workshop-Teilnehmern wurden Fragen, Ideen bzw. allgemeine Anregungen aufgeworfen. Durch den Autor dieses Berichtes wurden die entsprechenden Wortmeldungen ausformuliert und einer ersten groben Klassifizierung unterzogen:

Erhöhung der Akzeptanz des Performance Engineerings im Projektalltag:

- Immer wieder wurde die Frage der Anwendbarkeit von Performance-Engineering-Methoden und deren praktikable Umsetzung diskutiert; dabei ging es auch um die Frage, wie entsprechende Leistungen verkauft werden können bzw. wie die Akzeptanz im Rahmen der Softwareentwicklung geschaffen werden kann.
- Erfolgsstories zu Projekten, bei denen die Anwendung des Performance Engineerings maßgeblich zum Projekterfolg beigetragen hat, sollten auch publiziert werden.
- Aktuelle Rahmenbedingungen (Zeit- und Kostendruck) im Umfeld von Softwareentwicklungsprojekten sind einer vorausschauenden Performance-Analyse zumeist nicht förderlich. Hier gilt es, Aufwände für das Performance Engineering mit entsprechenden Mehrwerten für den Kunden transparent in Verbindung zu setzen.
- Notwendiger Wandel des Performance Engineerings vor dem Hintergrund der immer kürzer werdenden Produktzyklen; dementsprechend ist die Anwendung zeitintensiver Methoden zum Performance Engineering nicht zielführend.
- Es sollte der Zusammenhang zwischen der Architektur auf der einen Seite und der Statik auf der anderen Seite dargestellt werden und möglichst „10 goldene Regeln für eine gute Architektur“ aufgestellt werden.

Im Rahmen des GI-PEAK zu bearbeitende Themenstellungen:

- Einigkeit herrschte zwischen den Teilnehmern des Workshops über die Zielstellung des GI-PEAK. Dementsprechend sollte sich der Arbeitskreis als Katalysator für die Umsetzung theoretischer Erkenntnisse aus dem akademischen Umfeld in die industrielle Praxis verstehen. Dabei sollten sowohl methodische, technologische und wirtschaftliche Aspekte berücksichtigt als auch der Einfluss der Ausbildung bzw. die Ausprägung entsprechender Berufsbilder im Umfeld des PE aufgegriffen werden.
- Das Performance Engineering sollte sich klar zu anderen ähnlich gelagerten Disziplinen abgrenzen, wie z.B. dem Traffic Engineering (hierzu existiert ebenfalls ein Arbeitskreis im Rahmen der GI).
- Der diesjährige Workshop beschäftigte sich primär mit der Performanceanalyse kommerzieller Hard- und Softwaresysteme, der Klassifikation potentieller Einflussgrößen auf die Performance bzw. dem Test/Benchmarking.
- Die Anwendung stochastischer Modelle im Kontext des Performance Engineerings schlägt sich derzeit nur unzureichend in den Beiträgen des Workshops nieder. Hier sollte durch entsprechende Beiträge eine Verbindung mit praxisrelevanten Performance-Problemen aufgegriffen werden.

Methodische und technologische Aufgabenstellungen:

- Wie kann mit divergierenden Zielstellungen im Umfeld des Performance Engineerings umgegangen werden. So harmonieren die verschiedenen qualitativen Eigenschaften des späteren Produktes, wie z.B. Performance/Effizienz und Wartbarkeit nicht miteinander, sodass ein entsprechendes Optimum gefunden werden muss.

- Neu zu entwickelnde Systeme sollten flexibel auf sich verändernde Performance Anforderungen eingehen können. Hier wurden auch die Möglichkeiten und Zielstellungen von „services on demand“ bzw. „performance on demand“ diskutiert und die in diesem Kontext notwendigen Verrechnungen angesprochen. Insbesondere im Umfeld sog. Serviceorientierter Architekturen (SOA) können gewaltige Herausforderungen für das Performance Engineering erwartet werden.
- Gestaltung von Service Level Agreements und Service Level Objectives mit Hilfe der Performance Engineering Methodik. (Berücksichtigung von ITIL – Information Technology Infrastructure Library, Service Delivery).
- Optimales Verhältnis zwischen den Aufwänden, die im Rahmen der Software-Entwick lung und der Hardware eingesetzt werden. Zum Teil kann es durchaus wirtschaftlicher sein, entsprechende Probleme mit Hilfe eines erhöhten Hardwareeinsatzes zu lösen.
- Kontextabhängigkeit des Performance Engineerings - kann bei dieser Themenstellung tatsächlich ein wichtiger Aspekt im Rahmen der System- und Softwareentwicklung sein. Aus Sicht der Teilnehmer des Workshops hängt dieses maßgeblich von der Art des zu entwickelnden Systems und den aus möglichen Performanceengpässen resultierenden Risiken ab.

Förderung der Zusammenarbeit im Umfeld des Performance- und Software-Engineerings:

- Die Bekanntheit des Performance Engineerings ist immer noch relativ gering, so existieren derzeit nur eine Hand voll von nationalen und internationalen Interessensgruppen. Selbst bei internationalen Konferenzen (z.B. der WOSP-Tagung [WOSP 2004]) gibt es zumeist nur um die 100 Teilnehmer.
- Noch immer fehlt im Internet ein Diskussionsforum für die Performance Engineering Community. Hier sollten sich z.B. allgemeine Hinweise zu PE relevanten Themen, Links zu weltweiten PE Quellen und FAQ wiederfinden.
- Um die Themenstellung des Performance Engineerings auch im Umfeld des Software-Engineerings zu platzieren, sollte mit den entsprechenden Communities zusammenarbeitet werden bzw. auch gemeinsame Workshop durchgeführt werden.

Neben den im Rahmen der moderierten Panel Diskussion aufgegriffenen Themen wurden auch während der Vorträge interessante Denkanstöße gegeben. Im Folgenden sollen aus der Vielzahl interessanter Wortmeldungen zumindest zwei Metaphern wiedergegeben werden, die auch auf dem Workshop zu lebhaften Diskussionen geführt haben.

„In der Raumfahrt kann man auch nicht einfach einen Rechner dazustellen, wenn die entwickelte Software einen höheren Ressourcenbedarf aufzeigt, als ursprünglich geplant“ *(sinngemäß: R. Gerlich)*

„Jedes Haus braucht sowohl einen Architekten als auch einen Statiker. Das Performance Engineering unterstützt im Umfeld der System- und Softwareentwicklung die Aufgabenstellung des Statikers, wobei die Rollen des Architekten und des Statikers auch zusammenfallen können.“ *(sinngemäß: A. Hennig)*

Zusammenfassend kann festgehalten werden, dass es in immer stärkerem Maße darum geht, Beziehungen zwischen Architekturentscheidungen und den daraus resultierenden Performanceeigenschaften des späteren IT-Systems (bestehend aus Hard- und Software) im Rahmen der System- und Softwareentwicklung zu erkennen und unter Zuhilfenahme entsprechender Modelle, Methoden und Tools, soweit wie aus wirtschaftlicher Sicht nötig (z.B. Risiken oder explizite Kundeanforderung), zu determinieren.

Ausblick

Alle, die durch diesen Workshopbericht neugierig auf die Arbeit des Performance Engineering Arbeitskreises (GI-PEAK) geworden sind, seien auf den im kommenden Jahr an der Fachhochschule Köln stattfindenden Workshop PE 2005 verwiesen. Vielleicht hat der eine oder andere Lust, seine Erfahrungen im Umfeld des Performance Engineerings im Rahmen eines entsprechenden Beitrags auf dem PE 2005 zu publizieren. Wem das ein wenig zu spät ist, sei auf die noch in diesem Jahr zum Thema Performance Engineering stattfindenden Workshops verwiesen. So findet im Juli der UKPEW (United Kingdom Performance Enginieering Workshop) an der University of Bradford statt. Auch die nächste Tagung der MMB-Fachgruppe sowie die nächste CMG-Tagung in den USA werden sicherlich vielfältige Aspekte des Performance Engineerings aufgreifen. Darüber hinaus findet im Jahr 2005 der 5. Workshop on Software and Performance (WOSP) auf den Balearen in Spanien statt - sicherlich aktuell das größte Ereignis, das sich speziell dem Performance Engineering widmet.

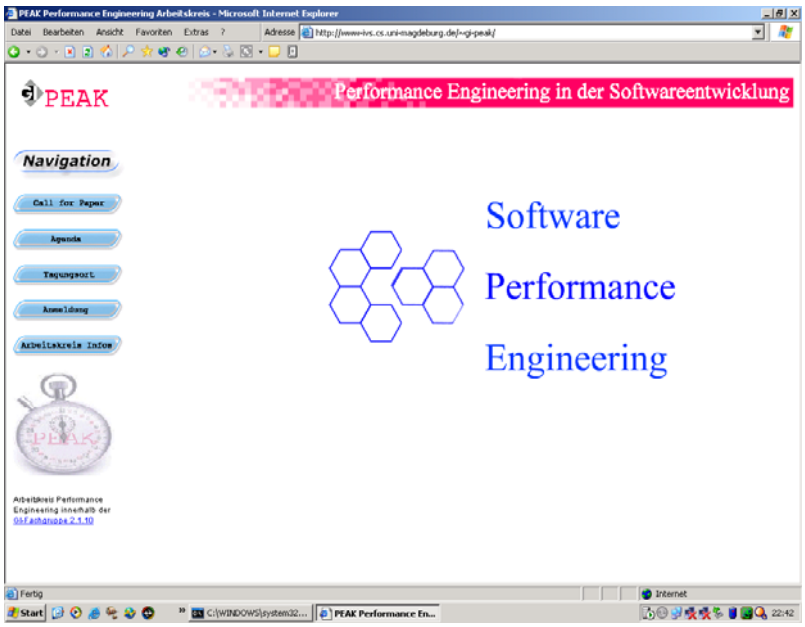


Abbildung 2: ivs.cs.uni-magdeburg.de\~gi-peak

Für weitere Informationen zu den o.g. Workshops soll an dieser Stelle auf die Webseite des Arbeitskreises (siehe dazu auch Abbildung 2) verwiesen werden.

Quellenverzeichnis

[Schmietendorf/Dumke 2004] Schmietendorf, A.; Dumke, R. (Hrsg.): Tagungsband PE2004, OvG-Universität Magdeburg, FIN, Mai 2004 (ISBN: 3-929757-65-6)

[WOSP 2004] WOSP'04 – Proc. Of the Fourth International Workshop on Software and Performance, Redwood Shores, CA/USA, ACM Press, Jan. 2004

Integration of software measurement into the software development environment

Bernhard Daubner
bernhard.daubner@uni-bayreuth.de

Andreas Henrich
andreas.henrich@wiai.uni-bamberg.de

Abstract: *Although the benefit of software measurement is commonly accepted, in operational practice one is often afraid of accomplishing the effort for the collection of software measures, because it seems to be out of proportion to the benefit. Therefore, software measurement should be automated as far as possible, which can be achieved by the integration of software measurement into the software development environment.*

This article describes two corresponding approaches on the basis of the development environments IBM Rational Suite and SAP R/3.

1 Tool support for software measurement

ISO/IEC 15939 [Int02] is the standard for software measurement published by the *International Organization for Standards* (ISO). This standard includes both a process model for software measurement and an information model. The later describes the information needs that lead to a choice of software measures and analysis techniques, whereas the information needs themselves are derived in turn from the goals of the software measurement. Thus, it is quite well known, which entities have to be measured in the context of the software development process to reveal the status of a project including possible risks.

However, we think there is a lack of appropriate tool support, that would automate the software measurement process as far as possible. Some measures, e.g. the number of open bug reports, are measurable quite easily in an automatic way. For instance within bigger projects one will certainly use a database for bug management and can therefore determine and observe very comfortably the number of bugs according to several categories.

The project progress in the sense of already implemented functionality of the product being developed can only be determined with a significant effort and only under certain circumstances. For this, it is necessary that one can trace back from the software component under development to the underlying functional requirement. From the development level of the related software components one can then deduce which functionalities are just under work, for which functions test cases exist and which software components have already passed the unit tests. So one gets a quite good picture about the implementation progress. At last, a requirement is classified *implemented*, if the corresponding software component has passed all functional tests and all integration tests.

In the following we will look at these possibilities with respect to their realization within the IBM Rational Suite and a possible integration into the SAP R/3 development environment.

2 Software measurement within the IBM Rational Suite

Within the IBM Rational Suite requirements are administered by *RequisitePro* within a relational database. Thereby the requirements can be linked with *Microsoft Word* texts or a UML diagram of the underlying use case. For the handling of test cases the Rational Suite includes the *TestManager*. This tool is able to generate test cases directly from requirements, UML models or Microsoft Excel spreadsheets.

Thus, within the IBM Rational Suite requirements are linked either directly or indirectly by means of a UML diagram to test cases. Additionally the requirements can be linked by means of a UML model with source code, which has been generated out of the UML diagrams.

IBM Rational Software has published a White Paper about the possibilities of software measurement that are provided by the IBM Rational Suite, especially if the tool *ProjectConsole* is used [GWI04]. Astonishingly, the software measures described within this paper are restricted to the mere enumeration of artefacts of the development process. For example one describes the number of requirements, changes in the number of requirements, the number of use cases with a certain status or things like that. Of course, one counts also the total number of lines of code and the number of bugs.

But there is nothing said about a possibility to directly determine the implementation progress from the software components under development. Corresponding software measures cannot be gathered until the programming activities have proceeded so far that particular use cases can successfully be tested against the developed software.

3 Software measurement with SAP R/3

3.1 Non object-oriented software development

Besides the object-oriented software projects there are also many development projects concerning legacy applications that either have to be maintained or enhanced. Here one might think for example of the huge host based accounting systems of banks and insurance companies that mainly are written in COBOL and do not have an object-oriented but at best a structured design. Besides there is a very large number of applications for SAP R/3, that are written in ABAP. Thereby ABAP has got an exceptional position, since with roots in the mainframe area it originally only supported structured programming, but has in the mean-while been extended with object-oriented concepts.

Especially as far as SAP R/3 is concerned there are hardly any scientific reports about the application of software measures within the SAP development process. There is only some work in which the requirements management in the SAP R/3 area is analyzed [Dan03]. Thereby also measures for effort estimation und software reuse are inspected.

Therefore, it is of great interest which software measures are actually applicable within ABAP projects, how the data for the software measures can be collected, and how this can be integrated into the SAP software development environment.

Moreover there is the question, which development process is actually appropriate for such an environment. From the authors point of view one might doubt, whether a use case driven process like the *Rational Unified Process* (RUP) is perfectly adapted for the creation of SAP applications, that are less characterized by user interaction but by mere data transactions. Here possibly a more adapted process could be derived from a process framework like *OPEN* [GHSY97].

3.2 Software organization within the SAP R/3 development environment

The role of the software repository within the SAP R/3 development environment is taken by a relational database system, that belongs to the R/3 system. This database system not only contains the business data of the R/3 system but also programs, input forms, table definitions, and data types. To access this data independently of the applied database system a general and system independent interface to this dataset is provided by the *Data Dictionary* [Mat02].

Besides the Data Dictionary also the *Correction and Transport System*, which is centralized in the *Transport Organizer* tool, plays a vital role within the SAP R/3 development. Beneath the version control the Transport Organizer provides possibilities to transfer software changes to other SAP systems and especially from the test system to the production system.

The actual organization of the software developed with SAP R/3 is done with *packages*, that were called *development classes* within former SAP releases. A package is a container combining development objects that belong together logically. Additionally packages can include subpackages.

The logic of an SAP application resides either within ABAP programs and subprograms, within function modules or within methods of *ABAP Objects* classes. Thereby, the classes provide the highest and the subprograms the lowest level of data encapsulation [Kel02]. A great part of present-day SAP functionality is indeed implemented with function modules [KK01]. Therefore, we want to present the integration of software measurement by means of function modules. These imply a very strict interface logic and are organized within function groups, whereby reusability and data encapsulation are supported at least to a certain degree.

3.3 Traceability and monitoring of project progress

In contrast to the IBM Rational Suite there is no integrated tool support for requirements management, modelling, and implementation within the SAP R/3 development environment. The requirements management and modelling activities are accomplished outside of the R/3 system whereas the coding and testing is done within the SAP R/3 development environment. In so far, there is first of all no automatic traceability guaranteed between the requirements, the development model, and the implementation.

Since there are, however, appropriate possibilities for the source code organization given within the SAP development environment, we want to present an approach, how this mapping can be done manually. As we have stated in [DH03], the UML activity diagrams are also suitable to represent data flows as they are used within the structured programming and especially within the modelling of SAP applications. Thereby, data flow diagrams (DFD) feature the possibility to adequately model the functionality of programs like SAP applications. Moreover, one can derive effort estimations from the DFD already at a very early stage of the project.

We propose to map the UML packages within which the activity diagrams are organized to the packages of the SAP development environment, whereas being restricted to only one lower level within the package hierarchy. One can then either associate each activity diagram with an ABAP function group and the enclosed activities with a ABAP function module, or the packages are structured in a more object oriented way combining all functions associated with certain data into one function group. At the end it is crucial to have a mapping between

the software model and the function groups and function modules. That means one must be able to derive from the name of an element of the software model the identifier of the associated function group and the function module respectively. If necessary the identifiers within SAP must be preceded with a prefix that is mandatory for customer developments.

By means of the Data Dictionary it can be checked which function groups and function modules have already been created within the SAP development environment, whereby at least a rudimental monitoring of the project progress can be realized. We want to show that this progress monitoring can be automated by determining the ABAP elements that belong to a software project via appropriate database requests. Because of their defined identifiers they can be assigned to the requirements modelled in the UML activity diagrams.

4 Progress measurement on an example application

Färber und Kirchner describe in [FK03] an example project, which deals with the implementation of an accounting application with ABAP on an SAP R/3 system. We want to use this example to explain our above ideas for progress measurement, because it is very accuratly described in [FK03] including business processes, architecture description and the resulting source code.

Within this example basically three functions have to be implemented that concern the warehouse and the accounting of a pharmaceutical company. Whenever a change in inventory within the warehouse takes place, a corresponding business transaction has to be stored in the database. These business transactions are from time to time transformed into accounting documents with respect to double entry bookkeeping. And finally an accounting list can be generated. Figure 1 gives an overview on this application.

Since the business transactions and the accounting documents are the two main elements of this application, it makes sense to pool them in two ABAP function groups with the identifiers `ZBTB00_OBJ_BTA` and `ZBTB00_OBJ_DOC`. Here `Z` is a valid prefix for customer developments within an SAP system, `BTB00` is an identifier for this application, and the authors of [FK03] use the identifier `OBJ` to characterize application logical functions in contrast to e.g. interface functions. Altogether, we get the SAP function modules listed in table 1.

Technically, a function group within the SAP system is nothing else than an ABAP program and a function module is an included subprogram. The main difference to normal ABAP programs is that function modules must not be edited with the standard editor but with the *Function Builder*. All function modules of the system are listed in the table `TFDIR` of the ABAP Data Dictionary. We now can query this table together with some other tables, that contain additional maintenance information, for all function modules belonging to our two function groups. Doing so we get the screenshot displayed in figure 2, where we can see that all above mentioned function modules at least already exist within our SAP system.

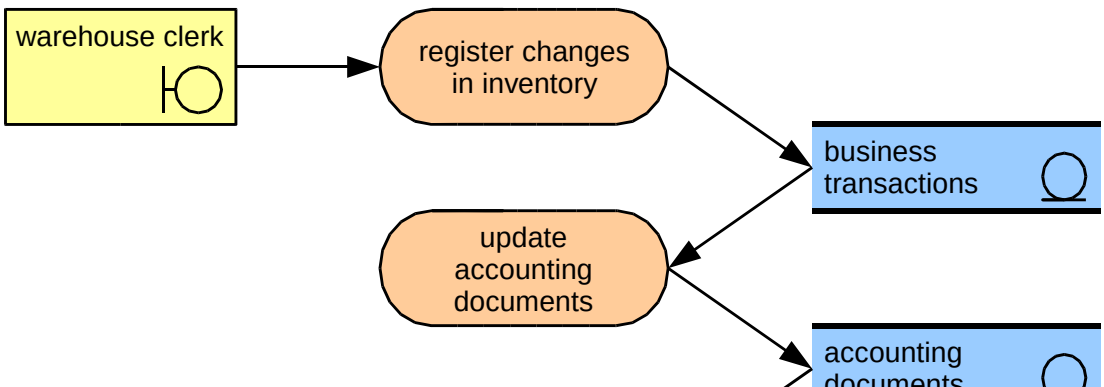


Figure 1: Accounting Application

ZBTB00_OBJ_BTA	
zbtb00_obj_bta_edit	edit business transaction
zbtb00_obj_bta_save	store business transaction into DB
zbtb00_obj_bta_load	get business transaction from DB
zbtb00_obj_bta_mark_as_booked	mark BT as booked within DB
ZBTB00_OBJ_DOC	
zbtb00_obj_doc_book	create accounting document from BT
zbtb00_obj_doc_save	store accounting document into DB
zbtb00_obj_doc_load	get accounting document from DB

Table 1: **Function Groups and Function Modules**

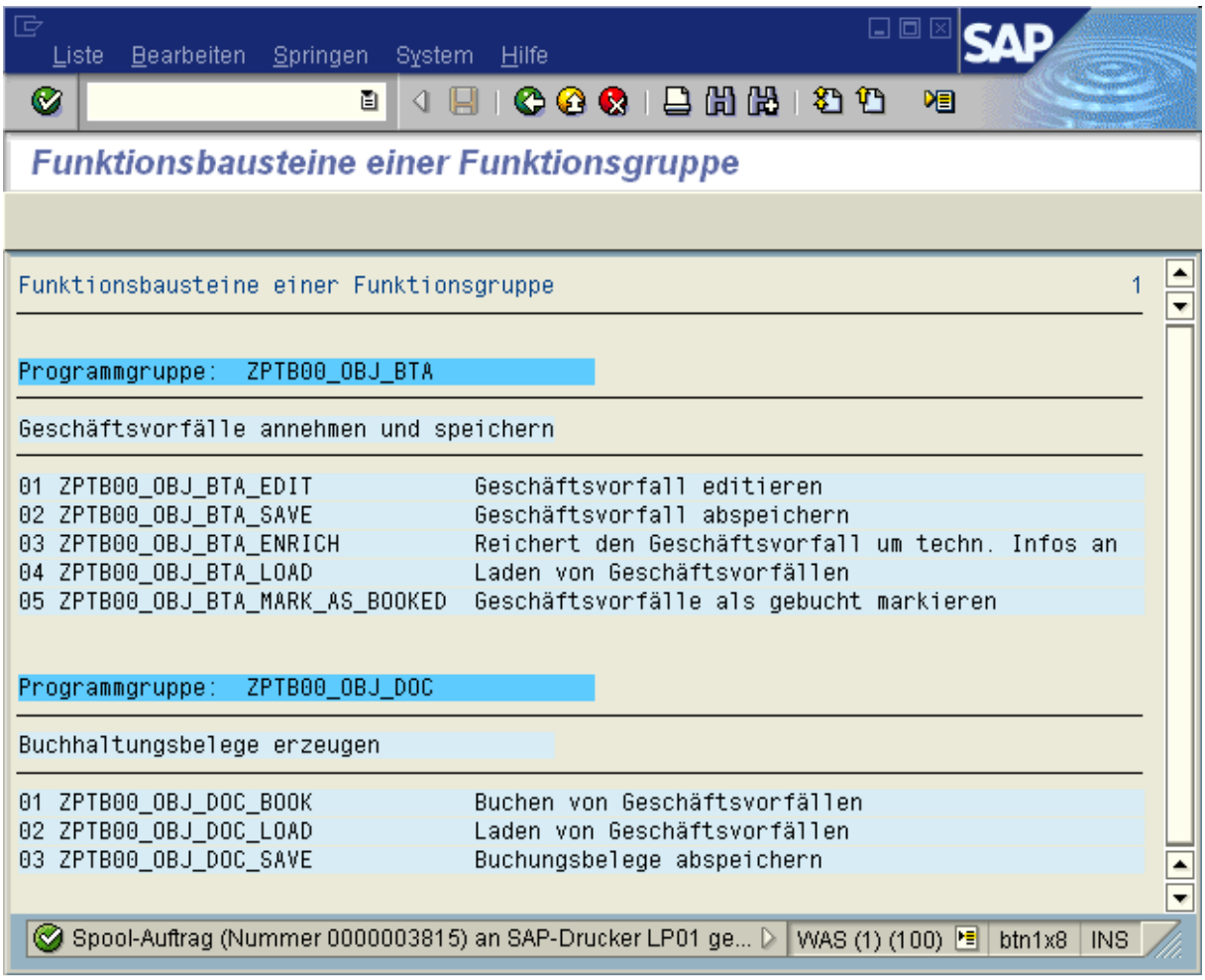


Figure 2: **List of function modules as a means of progress management**

What we have done hereby is to establish a link from the design model to the implementation by means of a strict naming convention. This offers a rudimental possibility to monitor the project progress, since one can check, which of the required functionality is already worked on. To provide complete traceability we also need a link from the requirements to the design model.

5 Further work to be done

So far we only know that someone has created the listed function modules in figure 2 within the SAP system. We do not know whether they work or even whether actually any programming work has been done. The next step could be to compute the sizes of code of these function modules that could possibly be compared with calculated Function Point measures.

We also have not yet considered, whether the corresponding entry masks to use this function modules have already been created. And we do not know, whether the database tables used by the function modules already exist.

The next step will be to provide a mechanism to derive these informations (definition of database tables, required forms) from the design model. Therefore, it is necessary that the modelling is done with an appropriate tool. Then this information must be read by an analysis

software within the SAP system to check how far the corresponding ABAP elements have been already created. The elements can be found because of their predetermined identifiers as we have shown above.

And finally a testing tool should be integrated into this framework to assure that the required ABAP elements not only exist but also work. Here the newly developed tool *ABAP Unit* might be a promising approach.

References

[Dan03] Maya Daneva. Lessons Learnt from Five Years of Experience in ERP Requirements Engineering. In *11th IEEE International Requirements Engineering Conference (RE'03)*, page 45ff. IEEE, 2003.

[DH03] Bernhard Daubner and Andreas Henrich. Ein Plädoyer für Datenflussdiagramme aus der Sicht der Aufwandsschätzung und der agilen Softwareentwicklung. In Klaus R. Dittrich, Wolfgang König, Andreas Oberweis, Kai Rannenberg, and Wolfgang Wahlster, editors, *INFORMATIK 2003 - Innovative Informatikanwendungen, Band 1, Beiträge der 33. Jahrestagung der Gesellschaft für Informatik e.V. (GI), 2003 in Frankfurt am Main*, volume 34 of *LNI*, pages 191–195. GI, 2003.

[FK03] Günther Färber and Julia Kirchner. *Praktischer Einstieg in ABAP Objects*. Galileo Press, Bonn, 2003.

[GHSY97] Ian Graham, Brian Henderson-Sellers, and Houman Younessi. *The OPEN Process Specification*. Addison-Wesley, Harlow (England), 1997.

[GWI04] Bill Gilbert, Dave West, and Doug T. Ishigaki. *IBM Rational Team Unifying Platform: IBM Rational ProjectConsole Sample Measures*. IBM Corporation, 2004.

[Int02] International Organization for Standardization. *ISO/IEC 15939:2002 – Software Engineering – Software Measurement Process*, 2002.

[Kel02] Rainer Kelch. *ABAP Objects – Ein Lehr- und Trainingsbuch für die klassische und objektorientierte Programmierung*. dpunkt-Verlag, Heidelberg, 2002.

[KK01] Horst Keller and Sascha Krüger. *ABAP Objects – Einführung in die SAP-Programmierung*. Galileo Press, Bonn, second edition, 2001.

[Mat02] Bernd Matzke. *ABAP – Die Programmiersprache des SAP-Systems R/3*. Addison-Wesley, München, fourth edition, 2002.

Estimations by Work Product Type: An extension of the UCP technique for the CMMI-SW level 2 and 3

Tatiana Cavalcanti Monteiro
Universidade de Fortaleza – UNIFOR
Mestrado em Informática Aplicada – MIA
Av. Washington Soares, 1321, 60811-341 – Fortaleza - CE – Brasil

Carlo Giovano S. Pires
Instituto Atlântico
R. Chico Lemos, 946, CEP 60822-780 – Fortaleza - CE – Brasil

Arnaldo Dias Belchior
Universidade de Fortaleza – UNIFOR
Mestrado em Informática Aplicada – MIA
Av. Washington Soares, 1321, 60811-341 – Fortaleza - CE – Brasil

Abstract. *The concern in generating quality estimations, the closest possible to reality, comes from the importance of these computations to estimate costs and time. Sometimes these estimations are made based on specialists' experiences, making it possible to generate inaccurate information. Besides that, there are methods to perform those tasks. One of those methods is the metric of Points per Use Case (PCU). This technique has shown itself adequate for object oriented software products and based on use cases. The CMMI-SW, level 2, recommends the implantation of size, effort, time and cost estimation activities, as a way of improving the planning and accompaniment of software products. However, this technique's granularity for work products in the planning and accompaniment activities hasn't shown itself very adequate. This work presents an extension of the UCP technique, so that it attends recommendations from CMMI-SW level 2, allowing a more detailed view from the estimations by type of work product, making it possible to refine those estimations throughout the development process.*

Keywords: *Estimation, UCP (Use Case Point), Use Cases, CMMI-SW.*

1 INTRODUCTION

One of the main risks that consummates the estimation process is the lack of credibility by the development team [10]. This occurs when estimations are unreal, that is the project are under or overestimated. The size estimations precision becomes fundamental for the elaboration of realistic schedule and budget, because the size estimations constitute the background for the derivation of effort, time and cost estimations [9].

One of the practices required by the model of CMMI-SW (Capability Maturity Model Integration for Software) level 2 is the realization of estimates for size, effort, schedule and cost. Some companies use proprietary methods to attend these practices, what makes very difficult the sharing of experiences, the use of knowledge bases outside the company, besides the effort and cost to define a proprietary method.

The estimates essentially support the planning and monitoring activities of a software project. Efficient estimates allow the verification of project viability, the elaboration of technical and commercial proposals, the development of plans and detailed schedules, and the effective monitoring of projects.

The estimates can be divided in two groups: bottom-up and top-down. The estimates bottom-up are used to achieve estimates about the items of work individually and then, summarize or aggregate them to obtain the complete estimate of the project. The estimates can be achieved through an expert's opinion or through the analogy of the database to determine the complexity and the effort associated to a certain task. The appraisal topdown estimates the project as a whole using specialized techniques. The granularity of the estimates of the work products is determined by the whole project's estimate. Information from analog products is used as base to the top-down estimate.

There are several techniques to estimate a software Project. Among them, some are more used than others, like: Functional Point Analysis (FPA) [7], MKII Function Points [7] and Use Case Point (UCP) [1]. The last one is based on the first two methods. In spite of the small divulgence, the UCP has been a goal to research studies and has shown growing utilization in the industry. The UCP technique is the top-down type and it is adequate to a project which describes its software requirements through use cases.

This work proposes an extension of the UCP technique – a Technical Use Case Point (TUCP) to achieve estimates of size and effort for software projects that are used as base for schedule and cost estimates, with the granularity proposed in the CMMI-SW level 2. Besides, the proposal suggests calibrations in the productivity factors by work product type, allowing better estimates for work products.

This work is organized in five sections. In section 2, we will describe how estimates are presented in the model CMMI-SW level 2 [9]. In section 3, concepts of use cases techniques are presented. In section 4, we present the description of the UCP method. We present our approach of estimates based on the UCP technique in section 5. In section 6, we present a case study for the extension of the proposed UCP method and finally in section 7, we draw conclusions to our work.

2 ESTIMATES IN THE MODEL SWCMMI LEVEL 2

CMMI-SW provides to software organizations a guide to obtain control in your software development and maintaining processes, and evolves in direction to a software engineering culture. The CMMI was projected to guide the software organizations in the improvement strategies selection process and identifying the most critical issues for quality and improvement of the software process [9].

Having in mind the reality and the organizations needs, and aiming to provide a larger flexibility, the CMMI adopts two approaches: one by staged representation, like the traditional CMM, and the other one continuous, corresponding to the ISO/IEC 15504 [11]. The staged representation comprehends five levels of maturity: 1-Initial, 2-Managed, 3-Defined, 4-Quantitavely Managed, e 5-Optimizing. The staged representation has the following level 2 (Managed) process areas: Requirements Management, Project Planning, Project Monitoring and Control, Supplier Agreement Management, Measurement and Analysis, Process and Product Quality e Configuration Management. The CMMI-SW continuous representation has six levels of maturity: Incomplete, Performed, Managed, Defined, Quantitatively, Managed, e Optimizing [9]. In the CMMI-SW and in the SW-CMM [9], the estimations process establishes a formal base for the planning and accompaniment of software projects, parting from four activities: (i) estimate the size of the product being generated; (ii) estimate the effort employed during the project execution; (iii) estimate the

project duration; and (iv) estimate the project cost. These activities are referenced in the CMMI-SW level 2 [9], in the Project Planning (PP) and Project Monitoring and Control (PMC) process areas.

In the CMMI-SW level 2 the size estimations need to be done by all the main activities and software products, and carried out with products with thin and adequate granularity for a continuous accompaniment. Nevertheless, this model doesn't specify the kind of measurement that should be applied, not even the granularity in which the work products should be decomposed. So that a project may be better estimated, their work products shall have to be decomposed until a necessary granularity to obtain the estimation purpose.

Next the points per use case technique (PCU) will be presented, that will be used as base for the project planning and accompaniment in organizations focusing on the CMMI-SW.

3 USE CASES DIAGRAMS

In Object Oriented Systems, use case modeling is commonly one of the first steps of the software development process. It's a technique used to describe and define the system's functional requirements. Use case diagrams represent the functional requirements. This representation discloses actors, use cases and their relationships. The actors represent the role of an external entity to the system. The use cases represent the system's functionalities or a classifier, like a subsystem or a class. The presented relationships may be of two types: association (between actors and use cases) and generalization (extensions and inclusions between use cases). The OMG is responsible for the formalization of these requirements modeling [12].

The Object Management Group, Inc. (OMG) is an international organization supported by over 600 members, including information system vendors, software developers and users. Founded in 1989, the OMG promotes the theory and practice of object-oriented technology in software development. The organization's charter includes the establishment of industry guidelines and object management specifications to provide a common framework for application development [12].

Despite the attempt of the OMG to standardize the functional requirements description in the use cases, organizations and project software teams don't follow it.

The concepts and bases of the standardized OMG use case techniques are indispensable factors in the UCP technique, acquainted in the following section.

4 THE METHOD OF USE CASE POINTS (UCP)

The object oriented software projects already use frequently the Use Cases Diagrams to describe the functionalities of the system according to the form of utilization by the users. Having as base the use cases, a technique for project estimate was proposed in 1993 by Gustav Karner [1], from Objectory (actually, Rational Software), allowing the possibility to estimate the size of a system, still in its use cases specification step, using its own generated documents as help for the dimensional calculation.

In spite of being a recent metric, the UCP has been studied by many researchers in the academic and industrial field. In [7], the practice results of the UCP estimating some commercial projects are mentioned. In order for this technique to be efficient, the use cases specification should be described in an appropriate detail level because this will influence directly the final quality of the measurement.

Once the system's main use cases are specified and described, it is possible to estimate the size of the whole software based on a simple metrics set. The necessary steps to generate the estimate based in UCP method are described below:

- Classification of Actors;
- Classification of Use Case;
- Definition of the Technical and the Environments Factors.

The summarize of this activities for generate the estimates are presents below.

The first step in the calculation of the system is to classify the actors involved in each use case, obtaining an unadjusted sum of points. The classification of the actors is based on Table 1. The total weight of the system actors (Unadjusted Actor Weight - UAW) is calculated by the sum of multiplication of the number of actors of each type by its respective weight.

Actor Type	Description	Weight
Simple	Application with defined API	1
Average	Another system interacting through a communication protocol, like TCP/IP or FTP	2
Complex	A user interaction through a graphic interface (*stand-alone* or Web)	3

Table 1: Classification of Actors

Once the weight of the system's actors is calculated, it's necessary to calculate the Unadjusted Use Case Weight (UUCW). For calculation purpose, the use cases are divided into three levels of complexity, according to the number of transactions involved in their processing. By transaction, we mean a series of processes that should be accomplished in a set or canceled in their totality, in case it's not possible to complete the processing.
The calculation of UUCW is accomplished in a similar way to the calculation of the actors' weight, adding the multiplication of the number of classified use cases in each type by its nominal weight.

Table 2 shows the weight for each one of the kinds of classified use cases.

Use Case Type	Number of Transactions Weight	Weight
Simple	Until 3 transactions	1
Average	4 to 7 transactions	2
Complex	More than 7 transactions	3

Table 2: Classification of Use Cases

To calculate the Unadjusted Use Case Points – UUCP as Eq. (1), it's necessary to add the value obtained by the actors measurement with the value obtained in the use cases measurement.

$$UUCP = UAW + UUCW \qquad \textbf{Eq. (1)}$$

The technical factors measure the complexity of a project regarding the non-functional requirements. These factors influence the result of the UCP as Eq. (4). According to Karner [1], the project complexity factors are the characteristics related to performance, portability, security, reusability of the code, among others (See Table 3).

Factor	Description	Weight
T1	Distributed System	2
T2	Response adjectives	2
T3	End-user efficiency	1
T4	Complex processing	1
T5	Reusable code	1
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portable	2
T9	Easy to change	1
T10	Concurrent	1
T11	Security features	1
T12	Access for third parties	1
T13	Special training required	1

Table 3: Technical Complexity Factors (TCF)

The environment factors are related with familiarity to the development process to be used in the project, the experience in the application, motivation, stable requirements, among others (See Table 4).

Factor	Description	Weight
F1	Familiar with RUP	1.5
F2	Application experience	0.5
F3	Object-oriented experience	1
F4	Lead analyst capability	0.5
F5	Motivation	1
F6	Stable requirements	2

F7	Part-time workers	-1
F8	Difficult programming language	-1

Table 4: Environmental Factors (EF)

To calculate the Technical Complexity Factor – TCF as Eq. (2) of the system, it's first necessary to calculate the Tfactor. The Tfactor is the sum of the multiplication of the value assigned in the project by each item and its weight. Calculation of the TCF:

$$TCF = 0.6 + (0.01 * TFactor) \qquad \textbf{Eq. (2)}$$

To calculate the Environment Factors – EF as Eq. (3), it's first necessary to calculate the Efactor. The Efactor is the sum of the multiplication of the value assigned in the project by each item and its weight. Calculation of the EF:

$$EF = 1.4 + (-0.03 * EFactor) \qquad \textbf{Eq. (3)}$$

To adjusted use case points (UPC) are calculated as follows:

$$UCP = UUCP * TCF * EF \qquad \textbf{Eq. (4)}$$

The Effort as Eq. (5) is obtained by the multiplication of the size in UCP by productivity that is calculated by:

$$Effort = UCP * Productivity \qquad \textbf{Eq. (5)}$$

The UCP as Eq. (4) count can vary among organizations and individuals because of the variation of use cases styles mentioned. It's reasonable to suppose that the productivity associated to the development of 1 UCP (20 men-hour, in Karner's original work [1]) varies a lot as well. This way, the obtainment of reliability estimates of effort requires style standardization of use cases in an extensive work of estimates model of calibration based on UCP.

As follows, the approach for estimations of work products based on UCP, the TUCP in the context of CMMI-SW.

5 AN APPROACH FOR ESTIMATES OF WORK PRODUCTS BASED ON UCP

According described in section 2, the CMMI-SW level 2 does not define which are the work products or the level of granularity necessary to reach the goals of the estimates. The companies are responsible for determining this level of granularity in their work products according to the objectives and necessary strategies of each organization.

The original UCP technique presents the estimate of size for the whole project in UCP and the estimate of effort can be obtained through productivity factors. However, this granularity does not allow a detailed planning, an effective monitoring and proactive of the project.

In this context, this work proposes an extension of the PCU technique, to attend to recommendations by CMMISW level 2, allowing a more detailed vision of the estimations by tip of work product, enabling refinements on the estimations throughout the development

process. This extension, the TUCP (Technical Use Case Point) comprehends the four following points:

- Project size estimation;
- Size estimations by work products;
- Effort estimation with differed productivity factor;
- Time and cost estimations.

The four points proposed by the TUCP technique shall be presented below, together with an utilization example.

Size Estimate of the project

Adjusted Use Cases Points – UCP as Eq. (4) consider the weight of the different kinds of interface through the actors, the weight of functional requirements through the use cases, the weight of the non-functional requirements through technical factors, and the weight of the staff and the development environment through environment factors. One problem identified with this approach is that the use of environment factors – EF can lead to different sizes, depending on the staff or company that develops the project. In fact, what should change should be only the effort according to the characteristic of productivity of the staff or company.

The proposed technique in this work is to define the Technical Use Case Point – TUCP as Eq. (6) metric to obtain a project size value based just on the functional and non-functional requirements of the system. This metric is obtained by the adjustment of UUCP by the Technical Complexity Factors - TCF.

$$TUCP = UUCP * TCF \qquad \textbf{Eq. (6)}$$

Size Estimates of Work Products

To accomplish the estimates of the work products, it's necessary to initially define the kinds of the product [9], generally a software development company works with four main kinds of products: Requirements, Analysis and Project, Implementation (Coding), and Test. One kind of software product can still be distinguished into sub-types.

For example, for the work product Analysis and Project, we can have the following sub-types: graphical project, architecture specification, detailed project, database specification and HCI (Human Computer Interface) specification for each use case. This way, the granularity becomes finer to estimate the use cases with easier accompaniment of the software project and allowing corrective actions before the end of all analysis and project.

According to the UCP method, the effort of the project is directly proportional to its size in use case points. This way, the size of a use case product can be based on the percentage of the effort put into its development (See example of measurements of the percentage of effort by the kind of product in Table 5).

Measurements accomplished for the Project, or for the kind of Project can be used to determine a percentage of adequate distribution for the institution. The size of each use case

product is then defined by a proportional form to its complexity factor and the percentage of effort distribution for each product.

$$\text{TUCP (use case product)} = (\text{Use Case Weight} / (\text{Sum of Use Cases Weight})) * \text{TUCP} * \text{PEA} \quad \textbf{Eq. (7)}$$

The work product size per use case, defined in Eq. (7) is given by the product of the division result between the weight of the use case and the total sum of weights of the system use cases by the size of the entire system – TUCP in Eq. (6) , which is multiplied by the effort percentage per product – PEP (see Table 5).

Effort Estimate with Differentiated Productivity Factor

To obtain the effort estimate, the UCP method uses a common multiplication factor (or productivity) for all kinds of project's activities. However the productivity can vary depending on the kind of project and the staff that accomplishes each kind of activity. For example, in a company that has frameworks or reusable components, the productivity of the coding activities are usually high.

However, the "specification of requirements" and "analysis and project" activities are not going to be fast because of the reuse of code. By the measurements of effort accomplished for each kind of activity and the TUCP's calculated for the project, the company should maintain a database with the productivity factor for each kind of activity and project characteristics.

The effort to generate a use case product is calculated by:

$$\text{Effort (use case product)} = \text{TUCP (use case product)} * \text{EF} * \text{PFKP} \quad \textbf{Eq. (8)}$$

The Effort defined in Eq. (8) consumed in a work product per use case is given by the product of the use case size – TUCP, presented in Eq. (7), times system's environment factor – EF, presented in Eq. (3), times the productivity factor of the work product type – PF (see Table 6).

Time and Cost Estimates

The time and cost estimates should be accomplished by the estimated effort with the proposed technique, resources availability, project restrictions and cost menhour. Notice that this estimate can be accomplished for the project as a whole or for each work product through the estimated effort for the product.

TUCP Example

To exemplify the technique, we are going to consider the use case Authenticate User as average, with a graphic interface (complex actor) in a company that has measurements and characteristics of project according to Table 5 and Table 6, a productive staff and a technical factor of medium complexity. Let's consider that there is also a use case Register User with Simple characteristics.

Product Kind	Effort Percentage
Requirements	20%

Analysis and Project	30%
Implementation	35%
Tests	15%

Table 5: Percentage of effort the Product - PEP

Factor	Value
EF	1.00
Sum of Weight: Weight*(Authenticate User)* + Weight*(Register User)*	15.00
TUCP of the whole system	18.36
PFKP - Productivity Factor (Analysis and Project)	15.00

Table 6: Productivity Factors -PF

With these characteristics, the total size of the project would be 18.36 TUCP's and for the use case we would have:

TUCP (Use Case Project *Authenticate User*) = (Weight (*Authenticate User*) / (Weight (*Authenticate User*) + Weight (*Register User*)) x (TUCP of the whole system) x (PEP- by Analysis and Project) = (10/ (10 + 5)) x 18.36 x 0.30 = **3.62 TUCPs**.

Considering the productivity factor of the activity of Analysis and Project as 15 (See Table 6) we would have an effort of:

Effort (*Authenticate User*) = TUCP (*Authenticate User)* * EF * PFKP (Anal*ysis and Project*) = 3.62 * 1 * 15 = **54.03 men-hour** for the *Analysis and Project* of the use case *Authenticate User*.

6 CASE STUDY

This case study was accomplished in a research and development company certified as SW-CMM level 2 that uses the process RUP (*Rational Unified Process*) [4] in research and development projects in many areas of information technology and telecommunications, in different platforms (J2EE, J2ME and .NET).

The estimate technique proposed in this article is applied to every software projects of the institution. For this case study, it will be presented two projects with different characteristics and size. The projects are denominated A and B for confidential matters.

The project A has as characteristic the development of a process automation system and was developed in the platform J2EE with 28.000 men-hours. The project B is a system for register and consult materials developed in the platform. NET with 2.000 men-hours.

The results presented in Table 7 and Table 8 show the percentage of relative error between estimated and accomplished values. For comparison effects, Table 7 presents estimated values with an only productivity factor for all kinds of products, while Table 8 presents the calibrated productivity factor for each kind of work products.

To calculate the percentage of estimated error for each activity using the proposed technique, the Symmetric Relative Error (SER) metric proposed by M. Jorgensen e D.Sjobeg [2] was used.

SER = Real - Estimated / Real, if Real =< Estimated

SER = Real - Estimated / Estimated, if Real => Estimated,

Where “Real” is the real effort of the Project and “Estimated” is the estimated effort using the technique proposed in this article.

In Table 8, the results presented show that the percentage of total error related to the real is much smaller if compared with Table 7, where there was not distribution of the productivity factor.

Project	REQ	A&P	COD	TEST	Total
A	-2.00	24.88	-49.03	29.08	-11.05
B	-47.81	-86.26	-32.96	-52.59	-46.61

Table 7: Percentage of errors without distribution of productivity by work product type

Project	REQ	A&P	COD	TEST	Total
A	-1.93	24.81	-11.76	29.08	3.24
B	-10.85	7.37	0.29	-34.61	-3.59

Table 8: Percentage of errors with distribution of productivity by work product type

The calibration on the productivity factor used in Table 8 considers that the factor of productivity in projects A and B were:

- Productivity Factor of requirements: between 15 and 20 men-hour.
- Productivity Factor of analysis and project: between 20 and 25 men-hour.
- Productivity Factor of coding: between 15 and 20 men-hour.
- Productivity Factor of tests: between 10 and 15 men-hour.

These values can be redistributed if the project and the staff’s characteristics influence some activity during the software development. According to Table 8, the values presented in project A for the products Analysis and Project and Test had a percentage of error bigger than the real.
I

n spite of the deviations in the percentage of errors in some work products being bigger than 20%, the total deviation in the effort is very small if compared with the individual values for each work product.

For Project B (see Table 8), the percentage of relative errors for the “Analysis and Project” and “Coding” were low. This happened because of the distribution of the main work products, that took into consideration that the “Analysis and Project” of a system of consult and register requires a smaller effort. During coding, the effort was also low because of the generation of the code that allowed a faster development.

Analyzing Table 7 and Table 8, we can verify that the distribution of the productivity by every kind of work product helped in the accuracy of the effort estimate. However, a historical base to be used in the calibration of the productivity is important because the factors as project characteristic, used platform and staff performance can influence this productive factor.

7 CONCLUSIONS

This article presented a technique for estimate of software based on UCP and compatible with CMMI-SW level 2. This technique, TUCP, can be used in projects that utilize use cases for specification of software requirements.

The main contributions to this work were:

- The extension of the UCP method through TUCP which allowed a more detailed view of estimates by kind of workflow;
- The productivity factor by work products type generates smaller error in the total estimate of the Project and in the estimate of each work products, making it possible to generate a more effective planning and monitoring;
- The proposed approach makes it possible to improve the estimates in many phases of the development process.
- A supply of an estimation technique adherent to the practices from the models CMMI-SW and SW-CMM [6] [9].

We can mention as the most important conclusions of this work:

- A historical base with the estimates of the company that should be implemented, in order to serve as grounding in the calibrations, in the factors of productivity for future projects;
- The inexistence of universal standards to the construction of use cases makes it difficult to compare projects from different companies. There’s no guarantee that the TUCP’s are going to be measuring the same thing, if the criteria used to build the use cases are very diversified;
- The use case elaboration should be described in an adequate detailed level, in order to the estimate based on UCP to be efficient.

REFERENCES

[1] Karner, G. **Metrics for Objectory**. Diploma thesis, University of Linköping, Sweden. No. LiTH-IDA-Ex-9344:21. December 1993.

[2] Ribu, K**. Estimating Object-Oriented Software Projects with Use Cases**. Masters thesis, University of Oslo. November 2001.

[3] Kruchten, Philippe. **The Rational Unified Process - An Introduction**. 2nd ed. New Jersey: Addison-Wesley, 2000.

[4] Rational Software Corporation, **Rational Unified Process**, version 2001.3, CD-ROM, Rational Software, Cupertino, Calif.:2001.

[5] Schneider, G., Winters, J. **Applying Use Case: A Practical Guide**. 2nd ed. Addison-Wesley, 2001. ISBN 0-201-70853-1.

[6] Paulk, M., Weber, C. **The Capability Maturity Model: Guidelines for Improving the Software Process**, 17th ed. Addison-Wesley, 2003. ISBN 0-201-54664-7.

[7] Symons, C.R. **Software Sizing and Estimating**, MKII FPA. Jonhn Wiley and Sons, 1991.

[8] Anda, B. **Comparing Effort estimates Based on Use Case Points with Expert Estimates**, 4th International Conference, Toronto, Canada, October 1-5, 2001, LNCS 218, 2001.

[9] SEI, 2002. **CMMISM for Systems Engineering/Software Engineering** ,Version 1.1-CMU/SEI-2002-TR-012. Disponível em: http://www.sei.cmu.edu/publications/documents/02.reports/02tr002.html. Acessado em 27/04/2004.

[10] Boehm, B., **Software Cost Estimation with COCOMO II**. Prentice Hall, New Jersey, 2000.

[11] ISO, 2003. **ISO/IEC 15504: Information Technology – Process Assessment**, Part 1 to Part 5.

[12] Object Management Group, Inc. **OMG Unified Modeling Language Specification**, Version 1.5 formal/03-03-01, March 2003. Disponível em: http://www.omg.org.

Software Assets Management – Modeling Issues and Proposed Models

David Déry
CGI
1350, boul. René-Lévesque Ouest,Montréal, Canada H3G 1T4
Tel: +1 (514) 415-3000x4946
david.dery@cgi.com

Alain Abran
École de Technologie Supérieure - ETS
1100 Notre-Dame Ouest, Montréal, Canada H3C 1K3
Tel: +1 (514) 396-8632
aabran@ele.etsmtl.ca

Abstract. *Too often, software intensive organizations can only track the initial assignment of a software to a resource but not necessarily thereafter. In such organizations, Software Asset Management (SAM) is often a reactive process. The lack of defined software asset management processes limits the ability of several organizations to manage the whereabouts of software once it is assigned to a resource. This puts the organization in a passive role so it is important to add planning and control processes, including for the retirement of software. To improve management of assets, the IT industry can learn from other disciplines, in particular from public works engineering. Through active assets management an organization will be better positioned to make choices to optimize and tune its Software Asset portfolio while complying with corporate policies.*

1 Introduction

In several software intensive organizations (SIO), a purchasing group handles software purchase orders. However, the lack of defined software asset management processes limits their ability to manage the software whereabouts. Too often, such organizations can only track the initial assignment of a software to a resource but not necessarily thereafter. In such organizations, Software Asset Management (SAM) is often a reactive (e.g. passive) process (see Figure 1): the purchasing group assigns the software to a resource (i.e. an individual, an organizational group or a server) and subsequently, on the basis of a pre-set contractual period, an invoice for a maintenance fee is received from a licensor/vendor and is paid. In such a reactive mode, decisions are taken one at a time, and the whole set of software assets is not managed from an integrated perspective: as a consequence, assets cannot be optimized and related maintenance costs cannot be minimized.

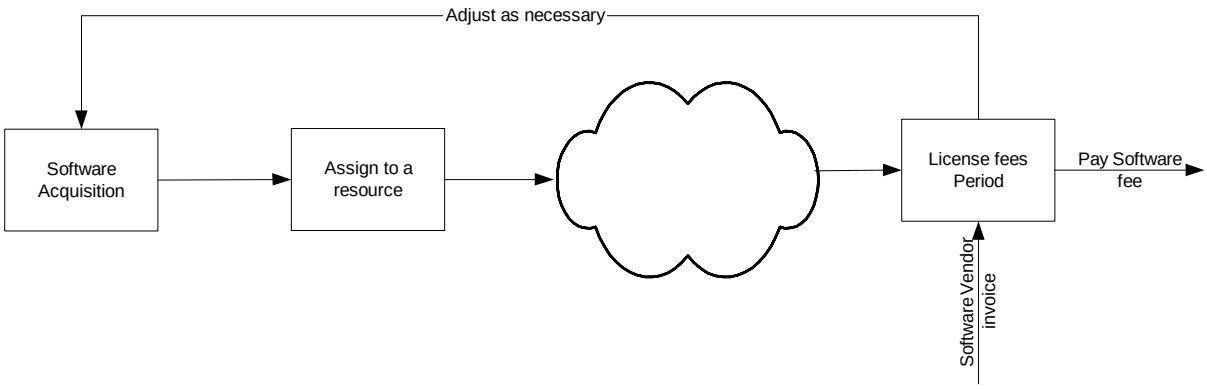


Figure 3: **Reactive asset management process**

This example illustrates the need to work towards a better understanding of the software asset management process and a better identification of the steps and external forces that influence these assets. Section 2 presents two related works: one found in the IS/IT industry and another found in public works engineering. In section 3, the methodology which is used to combine the two related works is described and explained. Section 4 presents the outcome of the combination between these two related works and finally, section 5 summaries the findings of this paper.

2 Related work

2.1 Information technology

Industry groups have proposed several best practice models and processes. In the field of information technology (IT), a set of best practices can be found in ITIL (Information Technology Infrastructure Library) [1]. ITIL is based on the collective experience of commercial and governmental practitioners worldwide and provides best practices for IT service management. It originated in the UK at the OGC (Office of Government Commerce) to address a high turnover of consultants. The OGC's motivation was to leverage the knowledge gathered by the outside consultants and capture this knowledge under the umbrella of a set of best practices.

This ITIL initiative is divided into two sections: Service Support and Service Development. The Service Support section identifies 5 processes and 1 service: Configuration Management, Change Management, Incident Management, Problem Management and Release Management processes and the service desk. On the other hand, the Service Development section identifies 5 other processes; Service Level Management, Financial Management for IT Services, Capacity Management, IT Service Continuity Management and Availability Management.

ITIL does not include asset management as a core process even though the need to interface with asset management is recognized. It is noted only that some organizations start with asset management before moving on to configuration management. This is because configuration management is considered to be a more complex process since the relation between assets are stored (i.e. technological dependencies), while asset management does not necessarily store this information.

Furthermore, in ITIL, the assignment process of software to a specific resource (Fig. 1) is part of release management with information about the software stored in the Definitive Software Library (DSL). This DSL, which, if properly maintained can be a good source of information for asset management, contains all the software (and versions of the software) in use.. In summary, ITIL does not identify asset management *per se* as a core process, the asset management process is not described and its components are not described anywhere in this best practices compendium.

2.2 Other disciplines

Other disciplines, such as public works engineering, have developed mature processes that are built exclusively for the management of assets. Some of these processes have become international standards such as the IIMM (International Infrastructure Management Manual) [2] which puts significant emphasis on assets planning.

At the heart of the IIMM is the Lifecycle Management Plan (LMP) that must provide background data on a variety of aspects such as Asset Capacity/Performance, Asset Condition, Asset Valuations and Historical Data.

Since IIMM focuses on planning, it includes several detailed plans: a Routine Maintenance Plan, a Renewal/Replacement Plan and a Disposal Plan. The Routine Maintenance Plan refers to the regular ongoing day-to-day work necessary to keep assets operating, including instances where parts of the asset fail and need immediate repair to make the asset operational again. The Renewal/Replacement Plan reminds the user that actions should be taken to ensure that the asset is either renewed (i.e. contractual) or replaced according to a pre-determined plan or agreement. The Replacement Plan is also required because if the asset is not renewed or needs to be replaced, a disposal plan should exist to explain how the asset will be disposed of.

3 Methodology used to build the model

As illustrated in figure 1, simply purchasing software and paying maintenance fees as bills are received is a very passive and reactive mode.

To be more proactive, planning is required. The IIMM applies these principles very well and it would make sense to apply these same principles to the management of software assets.

Figure 2 depicts (in the column on the left) what happens in the case of software purchasing: the software is ordered, allocated, recorded and a maintenance bill is received. This passive mode of operation has no planning and no control mechanism.

On the other hand, the IIMM spends a significant amount of time focusing on planning and outlining the importance of a good plan. But a plan is not of much use if it is not updated as required. This implies that there is a control mechanism to monitor and report on differences when they occur. This is illustrated in the right side of Figure 4, under “Engineering”. The differences identified by the control process help adjust the plan to better meet the corporate SAM requirements.

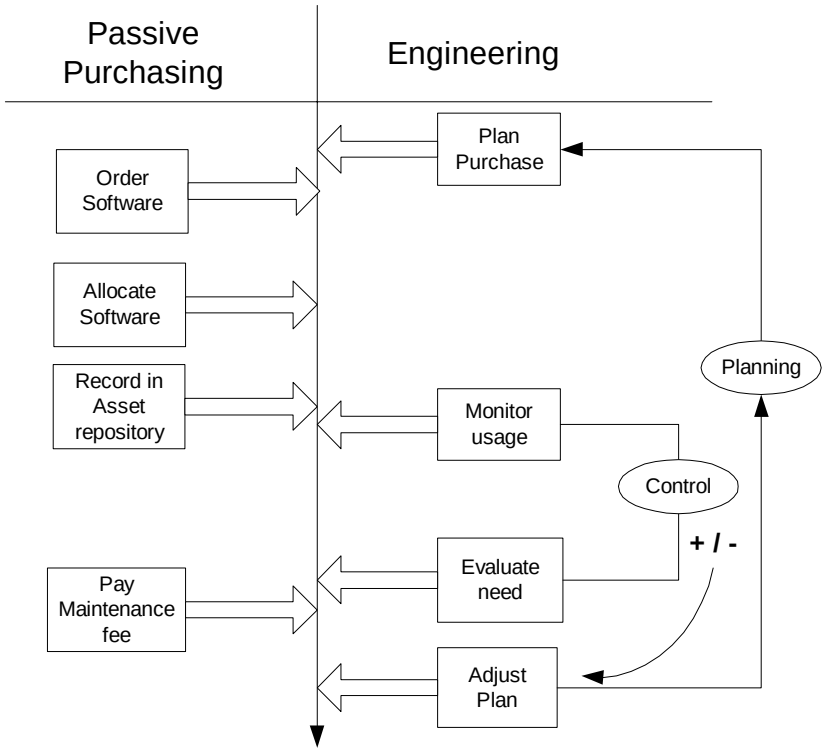


Figure 4: **Adding planning and control to software purchasing**

Adding planning and control to a passive process is only one element of the model. In the IIMM model, choices have to be made before adjusting the plan. These choices affect the very nature of the asset; to be operational, the asset must not only be maintained or upgraded but it may also need to be completely removed from the company's asset portfolio. Retiring software assets in a planned and controlled manner is not well documented in the IT industry whereas such a retirement process is quite common in public works engineering. To improve its SAM, the IS/IT industry can learn from public works engineering on how to plan for software retirement.

4 Proposed model

To provide adequate management of software assets, it is necessary that all relevant processes be included. Our proposed improved model of software asset management has been constructed by combining the strengths of both ITIL and IIMM frameworks. This approach has lead to the identification of a 5 step approach to SAM (see also Figure 5):

- Step 1: Corporate planning
- Step 2: Planning and purchasing of software
- Step 3: Assignment and monitoring
- Step 4: Reconciling needs and asset holdings
- Step 5: Asset portfolio tuning and optimizing

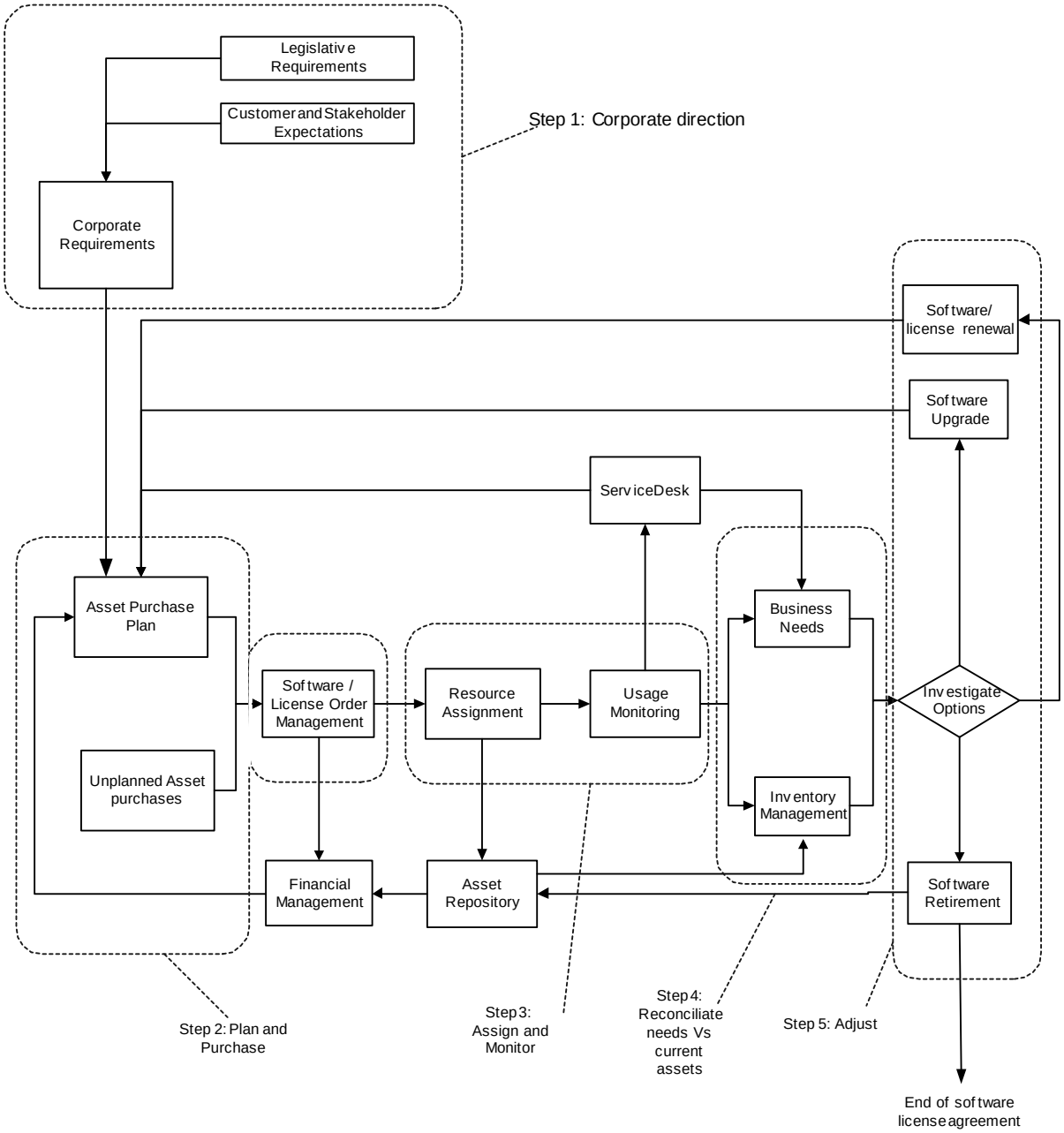


Figure 5: **Software asset management (SAM) process**

Step 1: Corporate planning

Planning plays an important role in the asset management process as is highlighted in IIMM [2]. It is important to decide upfront how much effort and budget will be assigned to asset management. This is the responsibility of senior management and the outcome is usually a tactical plan to help achieve the organization's long term goals.

This tactical plan plays a key role in determining corporate SAM requirements. These corporate requirements are also based on external input: it is important to take into account customer expectations (such as expected level of service and expected revenues from these services) as well as legislative requirements (such as financial and environmental constraints). This tactical plan, in addition to identifying how much to spend on software, will also specify how much formalism and tracking will be necessary to maintain control on software introduced and how it is to be used within the organization.

Step 2: Planning and Purchasing of Software

Guided by senior management input, an asset management plan is prepared to manage software purchases. It will feed purchase order management and provide guidance as to the type of software, the volume and the licensing scheme to buy.

In this planning process it is important to consider that in practice not all software purchases will have been included in the high-level plan, nor be fully aligned with the tactical orientations from senior management. Unplanned purchases may still be acquired in-between planning cycles but, once identified, must become integrated within the next asset purchasing plan.

Order management or purchasing is where the actual software purchase order takes place. These purchases will have a financial impact not only because of the purchase price but also because of the licensing costs which might include maintenance and upgrade costs. For this reason, it is important to feed IT financial management with any new licenses and contracts agreement with software vendors.

Financial Management as defined by ITIL [1] may include budgeting and IT accounting and charging. It is also the finance department that determines the budgeting rules and monitors and reports on the budget plans. It is therefore important to maintain alignment with the financial management process to ensure that purchases adhere to financial directives and that spending be kept under control.

Step 3: Assign and monitor

Once bought, the software is assigned to an owner-stakeholder: an individual or a corporate entity. Currently, this initial assignment is usually well recorded. However, any subsequent assignment to another individual or server may not be recorded. This explains how organizations risk losing track of the software. This inability to keep track of software might lead to unplanned overspending and at the end of the licensing period organizations then find out that they still are paying for a software they did not know they still had and, in many cases, that they might not be using anymore.
To minimize the risk of losing track of reassigned software, it is important to record any movement of software, server and related individuals within organizations. This tracking requires a formal asset repository where all information about the software, server and owner-

stakeholder is recorded. This repository bears some resemblance to the DSL described in Release Management of ITIL, but its content and level of detail must be aligned with corporate requirements.

Knowing who has a specific software and where it resides is, however, only part of the required information for SAM. Software vendors offer a variety of licensing schemes and determining which one is the most appropriate is not easy. This is where monitoring how the software is used can contribute and help make better decisions later on.

Furthermore, usage monitoring helps the service desk to determine the appropriate number of support staff to be assigned and to validate that the purchased licensing scheme is appropriate. Indeed, the service Desk as defined by ITIL is the single point of contact for customers and for operational needs to resolve incidents. This means that the Service Desk is also aware of software that causes the most problems and which ones are most requested for installment.

Step 4: Reconcile needs and assets holdings

Software licensing compliance is important but it should not be the only goal of software asset management processes. It must also include cost control to ensure that the appropriate license scheme is selected and is aligned with corporate objectives such as growth, flexibility and security. This means that the appropriate combination of quantity, license scheme, is purchased and maintained throughout the fiscal year with the right number of support people.

If software licensing compliance were the only goal, the organization might keep buying more and more software in order to avoid paying penalties for potential breaches of contract. When the organization is clearly buying too much to avoid non-compliance problems, the organization is paying more than the optimal amount because it lacks the information to determine the appropriate amount. To avoid this, software usage must be monitored and compared to business needs. A snapshot of current software asset is obtained though inventory management which can be conducted by monitoring the software used on a network and by performing scans on the network's computers to identify all software residing on individual computers (of course, additional procedures must be planned for computers which are not part of the scanned network). The list of software obtained through inventory management should be compared and matched to those in the asset repository. If a discrepancy is observed, corrective measures should be applied to reconcile the two views.

Aligning inventory management with the asset repository ensures that an organization knows what software it owns, but it does not tell about the adequacy of the licensing scheme, nor about the appropriate number of licenses required or even if the appropriate software is being used. For instance, business needs can be identified from corporate requirements and by analyzing what kind of calls the service desk receives for each software type. From this exercise, the company may need to make adjustments to its existing software portfolio.

Step 5: Asset portfolio tuning and optimizing

Once an organization has identified its assets portfolio, the question is what choices are to be made, and how to optimize and tune its Software Asset portfolio while complying with corporate policies. The decision for each individual software will usually be one of three major choices: keep the software (renew license), upgrade to a new version of the product or simply remove/retire the software and stop paying licensing costs.

Although there may be some variations, these three choices cover several cases. When a software is deemed satisfactory or if no alternative is found, this software is often kept and the licensing costs are renewed. If business needs or server requirements change, an upgrade is required and a new licensing scheme is usually necessary. Such upgrades occur following significant changes in requirements or business needs, and do not have to be with the same vendor. Finally, the software may no longer be needed and in order to stop paying, licensing fees must be retired. It is then particularly important to update the asset repository that, in turn, feeds financial management which pays incoming bills. This last item is often overlooked; when not properly managed organizations end up paying licensing fees for software they do not use anymore.

5 Summary and next steps

To better understand and identify what influences asset management processes and enable better software asset management, two related industry frameworks were investigated. By combining two such standards, ITIL[1] and IIMM[2], an integrated model was designed to include several enabling processes.

The next step includes validation of its content by experts who will verify completeness and relevance. Once this validation step is completed, it will then be tested in an industrial environment. This initial version of this asset management model is therefore subject to change and adjustments as more research is carried out and lessons are learned.

This model also addresses a need formulated by the industry and that is being worked on by ISO who is planning a Software Asset Management standard for 2006 [3].

References

[1] (OGC), "IT Infrastructure Library (ITIL)," 2001.
(see http://www.ogc.gov.uk/index.asp?id=2261 for details.)

[2] (IPWEA), "International Infrastructure Management Manual," 2002.
(see http://www.ipwea.org.au/ for details).

[3] (ISO/IEC), "Software Asset Management", TC JTC1/SC SC7/WG 21 WD 19770-1, International Organization for Standardization - ISO (Geneva), www.jtc1-sc7.com, Date: 2003-05-13 N006 ."

Quality Assurance of the project-related Software Development Process

Antje Riekehr
Otto-von-Guericke-University Magdeburg,
Fakultät Informatik, Institut für Verteilte Systeme,
Postfach 41 20, D-39016 Magdeburg,
riekehr@ivs.cs.uni-magdeburg.de

Andreas Schmietendorf
T-Systems International, Entwicklungszentrum Berlin,
Integration Services Wittestraße 30G, D-13476 Berlin,
andreas.schmietendorf@t-systems.com

Abstract. *Benchmarks are widely used to verify the maturity of project organizations. This paper shows our experiences with the implementation of a project related assessment. The assessment was driven from the wish to receive more transparency within an introduced project organization. We used as method for the evaluation our own benchmark process. This benchmark based on the identification of the process maturity, the realization of a strengths and weaknesses profile and the size measurement of the whole implementation. Based on the size measurement we derived the project related effort by the use of the COCOMO and Function Points method. Finally we compare the effort estimation with the real effort.*

1 Background and Motivation

The management and controlling of a complex software development project with several distributed teams is a very hard job. For the successful development of a software solution plays the quality of the underlying processes an important role. Quality aspects within the software development process deals not only with the quality behaviour of the product itself but also the qualities of all activities, that are necessary to the fulfilment of given requirements. This activities must be integrated in the process of the quality assurance during the whole time of the software development. [5] describes the integration of metrics in the software development as the *intelligence behind successful software management*.

Our goal was to implement a quality assurance process for a large software development project. This project deals with the implementation of a complex asset management solution. During the first version of the project the management team start with a chaotic process. In the first version it was important to reach a running system. Very often we can observe in early projects that the requirements are to complex. From our point of view it is important to find a pragmatic base.

- Identification of potential risks within the different project teams
- Effort estimation of specific development tasks
  - requirements engineering
  - design and implementation (divided in GUI and Kernel)
  - test and integration
- Implementation of a lasting improvement process
- Improvement of the communication culture in the project

The used benchmarking process and the interpretation of the results was carry out in cooperation with Software Measurement Laboratory of Otto-von-Guericke University Magdeburg and the Integration Services Group of the EZ Berlin/T-System International. [4]

2 Used evaluation process

The used evaluation process (benchmark) was developed on our own. In several projects, this already was applied successfully (see also [7]). This benchmark process based on experiences and also well established evaluation models. These evaluation models are the Capability Maturity Model (CMM) to benchmark the development processes and the Constructive Cost Model (CoCoMo, version II 2000) to measure the resources and the products as well as to post estimate the used effort of the product lines. Furthermore we estimated Function Points by the use of the backfire method. An other important part is the automatic source code analysis by the use of a tool. (see also [1], [2] and [3])

The whole analysis of the project subdivided into the following areas, which were worked off also in the following sequence.

1. Evaluation of the current situation in the project
   - Evaluation of the project documentation
   - Short questions to each members in the project
   - Use of external expertises (suppliers and customers)
   - Establishment of a goal driven procedure
2. Process assessment by the use of an adopted Capability Maturity Models (CMM)
   - Preparation of a corresponding questionnaire
   - Execution of structured interviews
   - Preparation of the interview results
   - Discussion of the reached results within a common workshops
3. Strengths and weaknesses analysis
   - Derived from the results of the CMM related interviews
   - Improvement potentials identify
   - Definition of a measure catalogue
   - Definition of measurable success criterions
4. Metric based analysis of the source code (e.g. LoC, Comments, used languages)
   - Use of measurement tools like RSM (Resource Standards Metrics) or others
   - Spot checks and estimations
   - Conclusions of the quality of the system (e.g.: reusability, maintainability)
5. Effort estimation
   - after the COCOMO II 2000 model
   - after Function Points (backfire method)
   - Comparison with the actual effort
   - Derives the own productivity

This here broadly described procedure, has to adjust for a concrete project in a suitable manner. In order to be able to receive diverse aspects of the project with help of the interviews, these should be prepared well. Results of the interview should cover general system requirements, answers for the CMM-related questionnaire and input parameters for the effort estimation. Furthermore it is important to discuss the expected goals of the benchmark with the project stakeholders.

3 Reached results

Within this section, selected results of the Benchmarks should be introduced shortly. The results refer to the initial application of the Benchmarks.

3.1 Evaluation of the current situation

Only by the knowledge of the actual condition of a project, possible measures can be introduced in order to improve this condition. For the analysed project we used also check lists to the investigation of the current state. These check lists contain statements about the product, to the resources (staff, hard- and software), to the process and to the requirements of the customers. Among other things following topics were taken into account within the first analysis:

- Project related topics:
  - Goals and content of the project: an asset management solution
  - Used programming languages and technologies

    GUI – ASP.net, C#, XML, Java Script

    Business components – J2EE, Java, XML, SQL
  - Degree of the automation: Model Driven Architecture approach
- Resource related topics:
  - Organization and structure of the team:

    Requirement engineering team

    GUI development team

    Business component development team

    Test and integration team

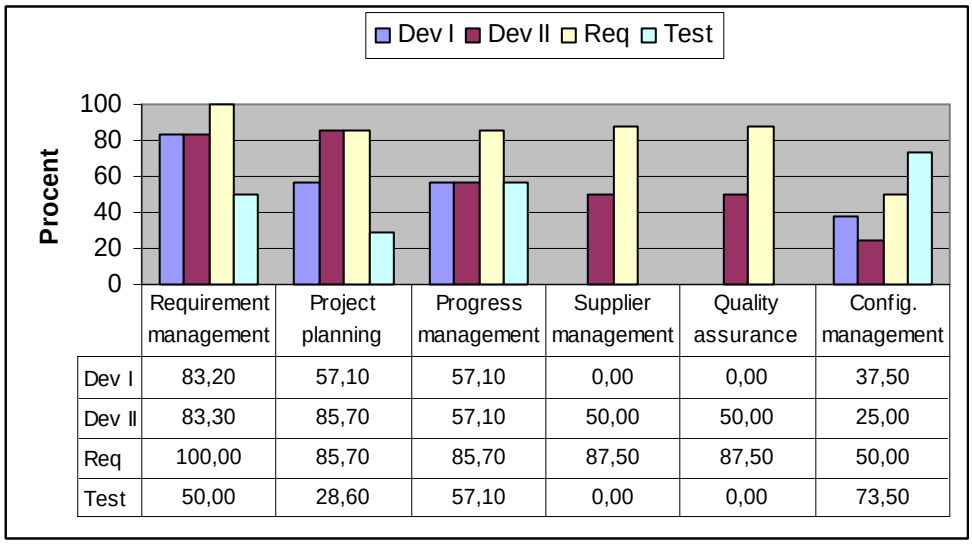
    Quality assurance team
  - Tasks and skill of the staff
- Development process:
  - Used process models: incremental and iterative
  - Identification of new requirements: by the help of a change request procedure
- Requirement Engineering
  - Functional requirements: use cases (order request, order information, …)
  - Non-functional requirements: concurrent users
  - Process- and system-related requirements: integration solution

3.2 Process evaluation with CMM

Within the evaluated project we used an adopted question catalogue under consideration of the CMM-level 2. Our own question catalogue covers the following main topics to reach the CMM-level 2. We used the question catalogue within our 4 project teams.

- Management of the requirements (6 questions)
- Planning of the software project (7 questions)
- Supervision and tracking of the software project progress (7 questions)
- Supplier management, like the used frameworks and other products (8 questions)
- Software quality assurance (8 questions)
- Software configuration management (8 questions)

To the achievement of the CMM-level 2 all 44 questions must be answered positively as well as with "yes". Figure 1 shows the results of the interviews with the 4 project teams.



	Requirement management	Project planning	Progress management	Supplier management	Quality assurance	Config. management
Dev I	83,20	57,10	57,10	0,00	0,00	37,50
Dev II	83,30	85,70	57,10	50,00	50,00	25,00
Req	100,00	85,70	85,70	87,50	87,50	50,00
Test	50,00	28,60	57,10	0,00	0,00	73,50

Figure 6: **Proportionally with "yes" answered questions**

Interesting is the very different assessment of the process maturity through the several teams. Absolutely typically, the very critical view is at the process maturity of the test team.

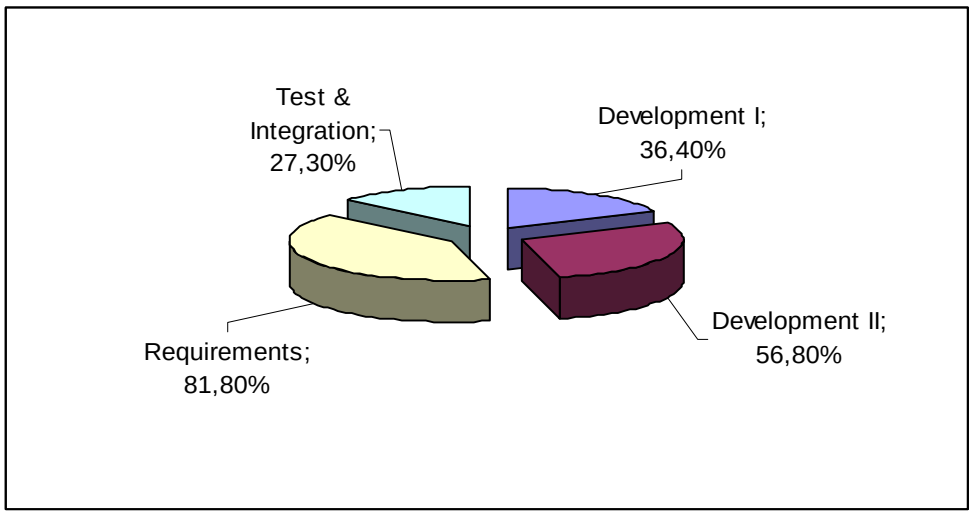


Figure 7: **Overall fulfilment degree after CMM (level 2)**

Under consideration of the interview results it is possible to derive a specific strengths and weakness profile for the evaluated project. Furthermore this profile allows the definition of activities to improve the process maturity under the consideration of project goals.

3.3 Strengths and weaknesses analysis

Within the executed interviews, the following strengths and weaknesses of the project could be identified. These were discussed within a workshop with all participants of the interviews. In the result, concrete measures (e.g.: procedure to deal with change requests) for the improvement of the process kindliness could be defined.

Identified strengths of the project (at the time of analysis):

- Estimations for the project planning are executed
- Tracking of the project through comparisons of the actual results and estimations
- Well defined project structure - responsibilities are clearly defined
- Activities of the configuration management are planed und executed
- Supplier management follows a selection procedure
- The project staff became well trained in accordance with her activities
- Correction measures are executed
- Results of quality evaluations are communicated to the project participant.

Identified weaknesses of the project:

- Difficult and partially unclear handling of change requests
- No periodic audits of the configuration management's contents
- Alterations of tasks to the sub contractors imply high risks
- Incomplete documentation of the project planning

3.4 Metrics based analysis of the source code

Another important part of the assessment was a metrics based analysis of the source code. These measurements offer an insight into the project to the management. In the following one, some selected measurements should be introduced.

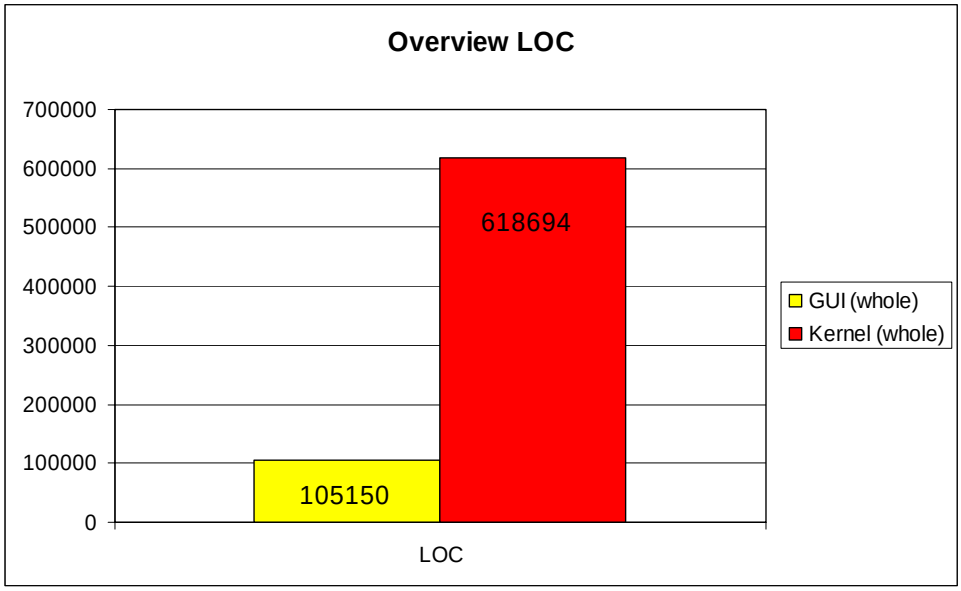


Figure 8: Overview about the whole project size in LoC

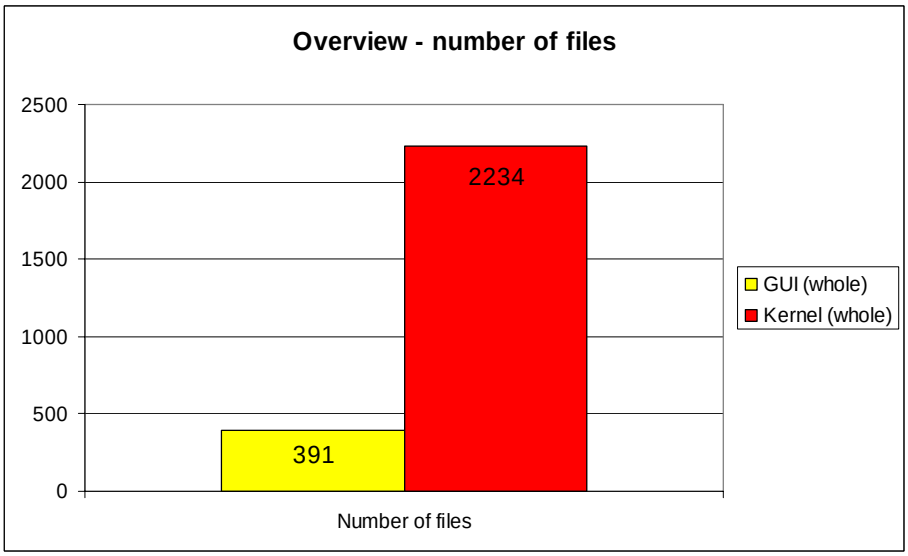


Figure 9: **Number of used files within the project**

Summarizing could be won the following information about the analysis of the source code.

- The share of generated commentaries within the GUI-implementation is 25 percent and within the KERNEL-implementation 22 percent.
- The used programming languages within the GUI-implementation covers:
  - C# - 74014 LoC (from it automatic generated 21689 LoC)
  - ASP.Net – 8340 LoC
  - XML – 3458 LoC
  - CSS – 784 LoC
  - JScript – 527 LoC
  - VBScript – 9 LoC

- WSDL – 18018 (from it automatic generated 18018 LoC)

o The used programming languages within the GUI-implementation covers:

- Java – 559044 LoC (from it automatic generated 533710 LoC)
- XML – 39336 LoC
- XSL – 3756 LoC
- SQL – 16558 LoC (from it automatic generated 918 LoC)

o The KERNEL-system contains following components:

- Activitivmanagement

  16497 LoC, 12550 eLoC, 8419 ILoC, 16531 comment, 40241 lines

- Delegate

  6880 LoC, 5755 eLoC, 2947 ILoC, 1145 comment, 9120 lines

- Exception

  25 LoC, 17 eLoC, 12 ILoC, 111 comment, 166 lines

- ProvisioningSystem

  14424 LoC, 10506 eLoC, 6788 ILoC, 6068 comment, 22717 lines

- Root

  655 LoC, 577 eLoC, 538 ILoC, 792 comment, 2201 lines

- Staffmanagement

  5545 LoC, 3970 eLoC, 2900 ILoC, 7260 comment, 15790 lines

- Stockmanagement

  14319 LoC, 10874 eLoC, 7448 ILoC, 19375 comment, 41062 lines

- Taskmanagement

  2633 LoC, 2105 eLoC, 1465 ILoC, 2681 comment, 6633 lines

- xCBL

  168708 LoC, 121434 eLoC, 91793 ILoC, 179479 comment, 398799 lines

- xCBL Validation

  75670 LoC, 61211 eLoC, 39407 ILoC, 20246 comment, 106582 lines

3.5 Effort estimation

The development effort were estimated with help of the COCOMO II 2000 (Constructive Cost Model) model. The COCOMO II 2000 model supports a fast and coarse estimation of the accruing efforts and if necessary the costs. The more exactly the result of the estimation should be, the earlier, in the development process, this should be executed. The result can be adjusted by the use of 22 influence sizes, 17 cost drivers and 5 scale factors. These required influence sizes were identified within the interviews. (see also [2])

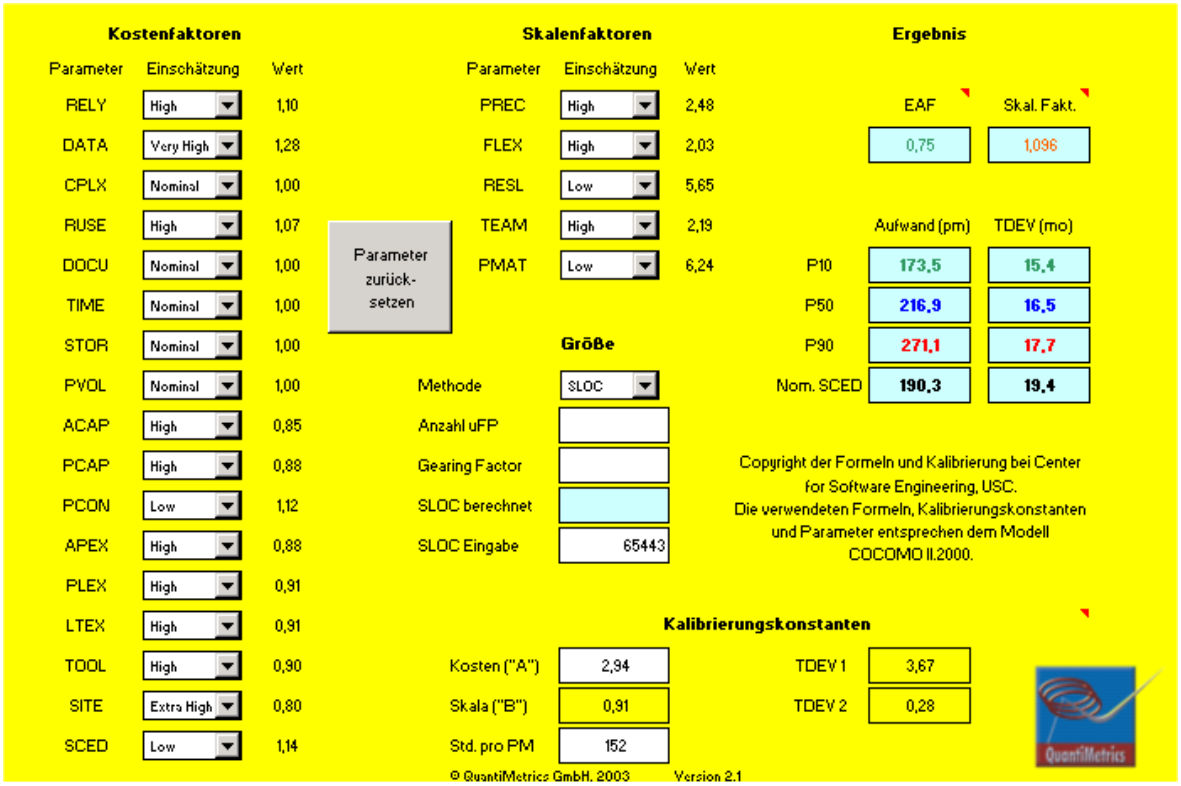


Figure 10: **Used tool to the calculation (Source: QuantiMetrics)**

Within the examined project, we have to consider a very high share of automatically generated source code. For the expenditure after-estimation the Excel-Tool "COCOMO_Calculator" was used. (Source: QuantiMetrics Ltd.). This allows the calculation of the required persons months, the calculation of the development time period and the number of required developers. The "COCOMO_Calculator" requires only the LOC, the scale factors and the cost driver as input parameters in order to calculate the wished efforts. The representation of the results took place in diagrams and tables. The estimated values were compared afterwards with the values of the real project and appraised.

For the comparison the number of developers and the time for the development are important information. All calculations of the project effort considers a fixed software version, therefore it was possible to compare the different implementations. The information about the real effort of the project were analyzed during the interviews with the project staff. Therefore we can examine that the time for development was 7 month. For the development of the user interface (GUI), 3 co-workers were appointed and for the KERNEL implementation 8 co-workers were appointed on average. Since the results of the COCOMO-calculation differed strongly from the reality, another after-estimation method were executed by means of Function Points. The Function Points were derived through the application of the backfire method. The backfire method based on the use of the „Gearing Factor“ and allows the calculation of Function Points (FP) derived from measured LoC [6]. By the help of estimated functions points it is possible to read the effort from available function point graphs.

In the following figures, the results of the COCOMO calculation are graphically represented:

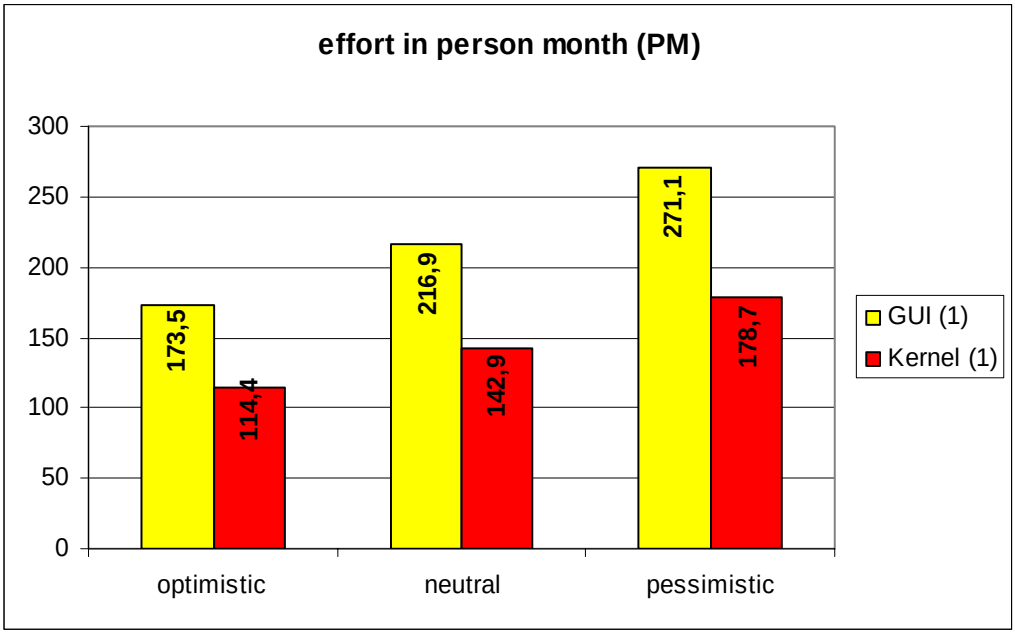


Figure 11: **COCOMO - Calculated effort in PM**

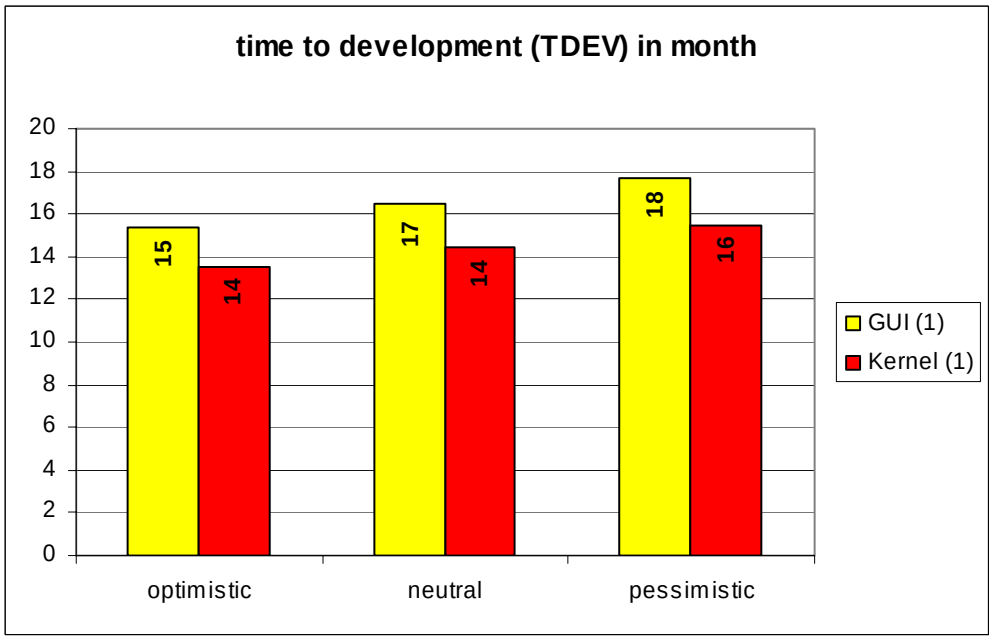


Figure 12: COCOMO - Calculated development time

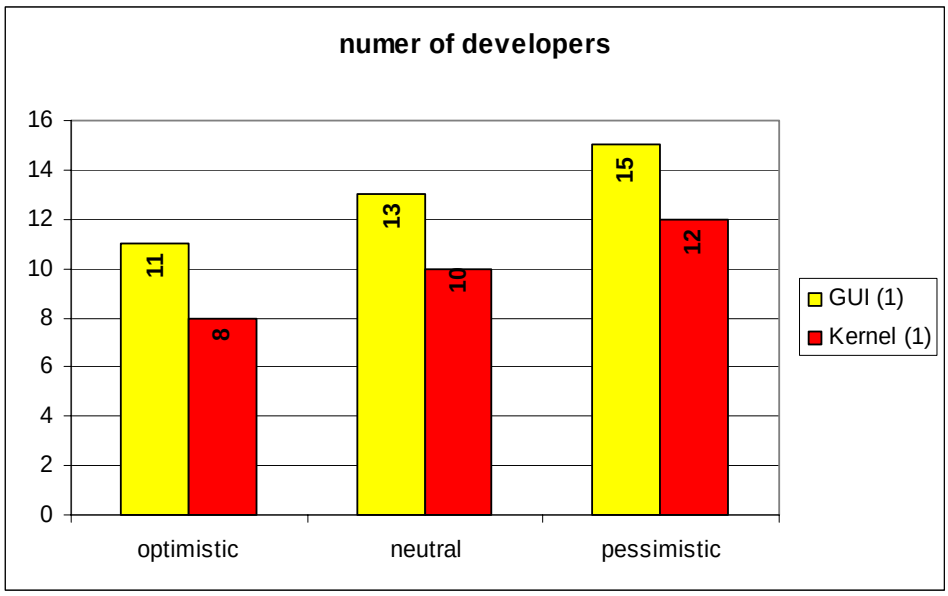


Figure 13: **COCOMO - Calculated number of needed developers**

4 Conclusions

It is recognizable that the results of the used estimation methods (COCOMO and also Function Points) differ strongly from the real effort. The real effort for the implementation in the project is less than the results calculated by COCOMO or Function Points. Also the real development time is shorter than the results calculated by COCOMO or Function Points. This result allows the conclusion that the productivity of the individual programmers significantly higher was than in other projects.

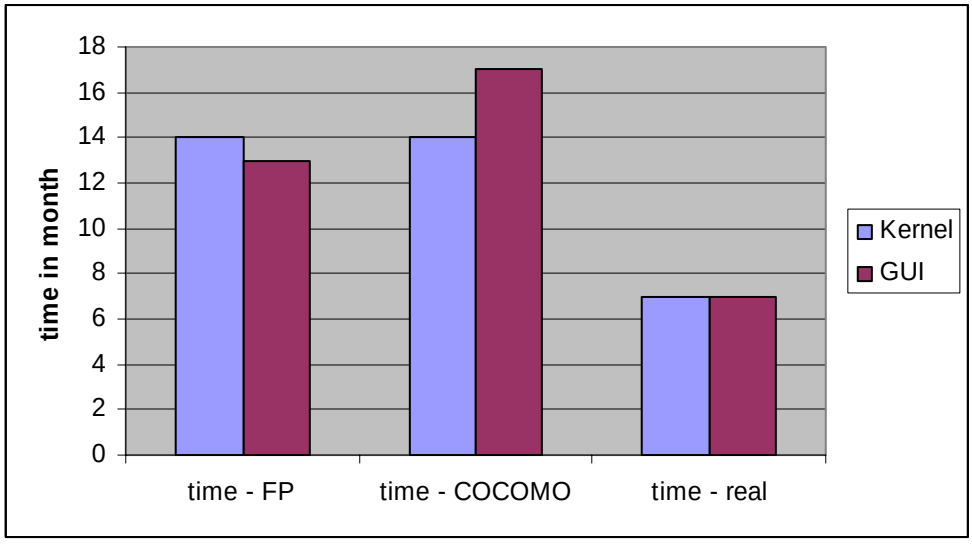


Figure 14: Comparison of the development time (FP/COCOMO/real)

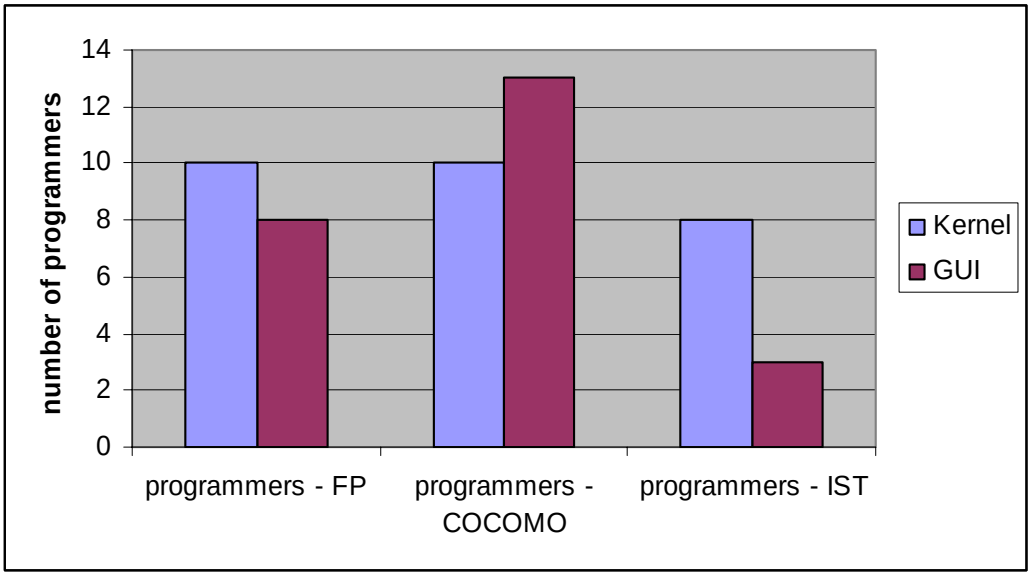


Figure 15: **Comparison of the needed programmers (FP/COCOMO/real)**

The development process of the project can be characterised as ad hoc. The costing, quality and development time is therefore unpredictable. From the negatively answered questions, the weak points of the process were determined.

The most important results of the assessment can be summarized as follows:

- Recognizes of potential project risks
- Prepares another view on the project
- Stimulates a discussion and communication between the project-teams
- Experiences with project sizes and resultant efforts
- Baseline for the process improvement

The described method is very useful for a project assessment during a running project. Furthermore it is recommended to realize an effort estimation at the beginning of the project by the use of Function Points, but not the here used backfire method. Original Function Points considers the required functionalities for the size measurement and not technical measurements like lines of code. In this way, the very high degree of automatically generated source code can not influence the result of effort estimation.

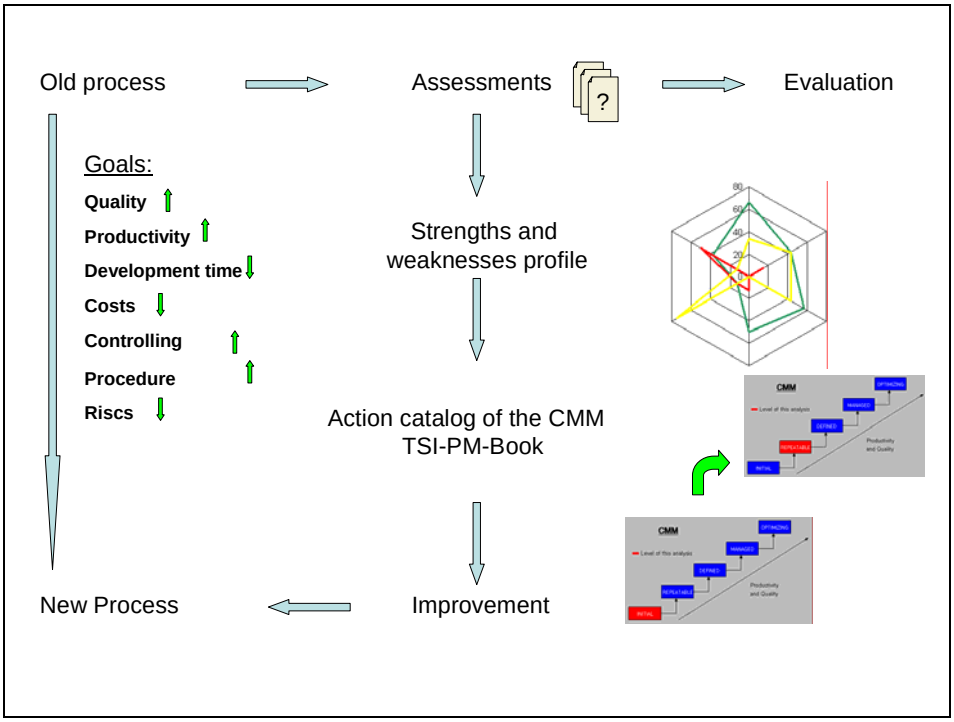


Figure 16: **Summary of the chosen procedure**

Next steps include also the repeated use of the evaluation model, an expansion of the assessment for other project types (e.g. integration project, introduction project) and the publication of the analyses within a web based portal.

References

[1] Ahern, D. M., Clouse, A., Turner, R.: CMMI Distilled – A Practical Introduction to Integrated Process Improvement. 2nd edn. Addison-Wesley, Boston (2003)

[2] Boehm, B. W. et al: Software Cost Estimation with COCOMO II. Prentice Hall Inc. (2000)

[3] Reiner Dumke, „Software Engineering“, 3. Auflage, Vieweg Verlag Braunschweig/Wiesbaden (2001)

[4] Ebert, C.; Dumke, R.; Bundschuh, M.; Schmietendorf, A.: Best Practices in Software Measurement - How to use metrics to improve project and process performance, Springer, Berlin Heidelberg New York (2004)

[5] Putnam, L.H.; Myers, W.: Five Core Metrics – The Intelligence behind successful software management, Dorset House Publishing, NY/USA (2003)

[6] QSM. `http://www.qsm.com/FPGearing.html`. downloaded 21.07.2004

[7] Reitz, D.; Dumke, R.; Schmietendorf, A.: Metrics based comparison of project lines in the industrial software development, in Dumke, R.; Abran, A. (Hrsg.): Investigations in Software Measurement S. 131-143, Shaker-Verlag, (2003)

An approach to a data oriented size measurement in Software-Product-Families

Sebastian Kiebusch
Universität Leipzig
Wirtschaftswissenschaftliche Fakultät
Institut für Software- und Systementwicklung
Professur für Wirtschaftsinformatik, insbes. Informationsmanagement
Marschnerstraße 31, D-04109 Leipzig
kiebusch@wifa.uni-leipzig.de

Summary. *This elaboration describes the adaptation of the first three Function-Point steps as a partial approach to estimate the effort in Software-Product-Families. The examination is based on general Product-Family requirements to a proceeding of cost estimation with a view to generative programming.*

Introduction

A SPF is a "… collection of products that share common requirements, features, architectural concepts, and code, typically in the form of software components" [1]. This modern software engineering paradigm is a promising solution for the current requirements of software products, consisting of high functionality and flexibility in combination with low costs.

The requirement of a holistic realization of process focused commonalities and variabilities is based on cross branch workflows of organizations in a dynamic and global market [cf. 2]. Therefore a lack of combination between the synergetic areas of Workflow-Management and SPF is illustrated. In addition, models to estimate the effort for projects and products in process oriented SPF are necessary.

Function-Point oriented methods for effort estimation are based on an indirect cost evaluation by accessing the size of a software system. The starting point of this size measurement is the requirement specification which is translated into function points. Empirical data of process directed SPF do not exist. In consequence the abstract proceeding of size measurement by the International Function Point User Group (IFPUG) as you can see in [3] is suitable for an accommodation to the paradigm of SPF oriented software engineering.

This new engineering paradigm in combination with generative programming and domain specific languages embrace fundamentally shifted cost structures. As a result there are different requirements to a cost estimation model for process focused SPF:

1. Type of count: You have to differ between effort estimation for a product or project in a SPF which is to develop or to modify or to reuse.

2. Time of estimation: Early forecasting because the majority of financial outlay fall on to the initial steps like domain scoping, analysis and modeling.

3. Information keystones: The useable data base is restricted to the declarations in the requirements specifications and to the facts from the Scoping-Product-Map.

4. Commonalities: The reuse of components in every product lead to a decrease of development costs despite increased demands of quality and compatibility.

5. Variabilities: Product individual elements increase the total effort for generating variants of applications in a SPF.

6. Process complexity: Variant dependant aspect of effort within the framework of construction and maintenance in process focused SPF.

7. Quality: You need to consider the outlay for realizing the necessary requirements of quality by the quality model which is standardized in [4].

8. Structure: Explicit separation between the efforts of products, projects and SPF for a well structured cost management.

The identified requirements put a strong request at a meta method for effort estimation in process oriented SPF. Furthermore the described requirements are the partial principles for the following adaptation of the first three steps from the Function-Point-Analysis.

Type of count

In relation to the first requirement you have to distinguish between the following types of counts at the beginning of the meta method:

- Development project count: Delivered functionalities which are ready for use by the consumer after the development of a SPF and the generation of a product variant.

- Reuse project count: Functional size of a product which is generated out of an existing SPF. In addition you have to differentiate between complete (SPF utilization) and partial (SPF modification) functional covering of a variant by the SPF.

- Application count: Measure of the actual provided functionality which is directly attached with the installed product and initialized after the project.

Figure 1: Types of counts for SPF and its relationships

Figure 1 graphically describes these relationships of the types of counts for SPF based on three alternative projects. All scenarios of the illustration support the consideration of additional functionality that was not specified in the requirements but identified during development (scope creep). Furthermore the construction of an empirical keystone is facilitated by repeated counting and the documentation of this act. Within the framework of

an entire functional covering between the product and the SPF you can exclude the phenomenon of the scope creep. For that reason the execution of a second calculation is in project C obsolete and therefore coloured grey in figure 1.

Counting scope and application boundaries

With dependence on [3] you have to treat the counting scope and the application boundary differently like it is described in the following items [cf. 5]:

- Application boundary: Distinction among internal and external functionalities as well as demarcation of the software which then is measured.
- Counting scope: Application independent border which can be embrace more or less functionality as a single software program.

Figure 2: Counting scope and boundaries of the SPF

In consideration of the eight requirement a distinction between the counting scope of the SPF and the generated product is necessary like shown in figure 2.

An additional distinction is based on the fourth and fifth requirement and is situated in the SPF as well as in the single product. Correspondingly you have to differentiate between commonalities and variabilities by virtue of unequal effort outcomes. Furthermore there is a need to assign the variabilities to the appropriate product variants of the considered SPF.

Count data functions

In addition to the locality it is important to regard the reuse of the counted data function because of the fourth and fifth identified requirement. According to table 1 there are four data functions to measure projects and products in SPF.

Table 1: Data functions to measure the size of SPF

The typification in table 1 is based on the following assumptions:

- External maintained files are referenced through internal interfaces.
- Contemplation of logical coherent data from the perspective of the customer.

The traditional Function-Point-Analysis includes a general accepted and historical grown method to determine the functional complexity of data assets. A detailed description of the captured IFPUG- complexity identification is explained in [3].

The following transformation of the DVE and DVI rely basically on the origin conversion factors of the Function-Point-Analysis. The reason for this is that an equal implementation effort between variabilities and traditional developed components is assumed. These individual components are characterized by a reuse in reliance on their product independent implementation frequency (IH, *germ. Implementierungshäufigkeit*).
Within the framework of keeping the high requirements of quality and modular interfaces for a generic component implementation, the DGE and DGI are especially critically. Therefore the complexity dependent conversion factors for commonalities in SPF are higher than their pendants for the transformation of variabilities.

There is a need to embrace the reuse of DGE and DGI in every generated product of a SPF. The absolute amount of effort decrease behaves itself proportionate to the number of generated products (PA, *germ. Produktanzahl*) in a SPF.

Complexity dependent correction factors for variabilities (KV, *germ. Korrekturfaktor Variabilität*) and commonalities (KG, *germ. Korrekturfaktor Gemeinsamkeit*) supplement the

conversion factors. They enable the consideration of historical experiences and influences of a SPF orientated development. Components with a high degree of complexity profit above average from the visual support of domain specific languages and code generation. Consequently the KV-/ KG-Values for high complex components in a SPF are lower than the correction factors for less complex components.

Finally for a definite differentiation of four data functions in three complexity mouldings, you have to enlarge the original IFPUG- conversion factors by two new values. Furthermore it is necessary to reflect the authentic conversion factors {5; 7; 10; 15} in [3] as a sequence of numbers by the following linear independent, cubic function:

By utilization of this third degree function it is possible to calculate 23 and 35 as additional conversion factors. At this point it is possible to measure the size of variabilities in units of unadjusted Process-Family-Points (PFP)s by using the next function:

In addition the following formula determines the unadjusted PFP for commonalities:

The final outcomes of the explained examinations regarding data functions in SPF are summarized in table 2. Here you see twelve transformation quotients which take account of historical experiences (KV/ KG), reuse (IH/ PA) and complexity (low/ medium/ high) as well as locality (external/ internal) of data functions in a SPF.

Table 2: Transforming complexity weighted data functions

For a hypothetical, high complex DGI which is part of a SPF consisting of four products and characterized via an empirical KG of 8/10 you will measure seven unadjusted PFP. The next equation describes the calculation for this theoretical example:

Conclusion and outlook

The explained method, referring the first three Function-Point steps, realizes an unadjusted, data oriented size measurement of product variants in SPF.

Figure 3 illustrates the entire PFP concept to estimate the effort for process oriented SPF in multiple domains. The research work for every dark grey coloured component is published in this article. Each module with a light grey emphasis is also concluded in terms of investigations for the present moment [cf. 6]. The derivation of a micro analysis for technical domains is under development and will be supplemented by the other white sections which are focused in future scientific exertions.

Figure 3: The PFP approach to estimate the effort in process oriented SPF

Necessity of additional research exists in terms of regarding generalized aspects of ISO/ IEC 14143 and different features of methods that are derived from the original Function-Point-Analysis.[1] At this stage the development of the approach for effort estimation in process focused SPF could result in a developer perspective and the consideration of characteristics from real time applications.

The total absence of empirical data is the main problem in further activities of investigation. A hypothetical SPF derived from the domain of automotive will support the solving of this difficult situation. In addition to this attunement the derivation of a regression function is required to estimate the effort of products and projects in process oriented SPF.

In the end the final effort estimation system must cover all the identified requirements of process focused SPF as explained in chapter one of this paper.

References

[1] Riva, C., Del Rosso, C. Experience with Software Product Family Evolution. In: Proceedings of the sixth International Workshop on Principles of Software Evolution (IWPSE'03), Helsinki, September 2003.

[2] Scheer, A., W. Wirtschaftsinformatik: Referenzmodelle für industrielle Geschäftsprozesse. 7. edition, Berlin 1997.

[3 The International Function Point Users Group (Ed.). Function Point Counting Practices Manual: Release 4.2, Clarkston 2004.

[4] International Organization For Standardization/ International Electrotechnical Commission (Ed.) Software engineering – Product quality – Part 1: Quality model. ISO/ IEC 9126:2001(E), Geneva 2001.

[5] Bundschuh, M., Fabry, A. Aufwandschätzung von IT-Projekten. 2. edition, mitp-Verlag, Bonn 2004.

[1] An intensive investigation and partial adoption of the COSMIC Full Function Point method is planned as well as a narrow study of the following approaches: Mark II Function-Point-Analysis, NESMA Functional Size Measurement, Function Bang, (SPR-) Feature Points, 3-D Function Points, Data Points, Object Points and Widget Points.

[6] Franczyk, B., Kiebusch, S., Werner, A. Stakeholderanalyse im Scoping- Prozess sowie Metriken der Umfangsmessung von prozessorientierten Softwareproduktfamilien der eBusiness-Domäne. PESOA- Report No. 08/ 2004, Universität Leipzig, October 2004, to be published by www.pesoa.org.

SOFTWARE ENGINEERING ONTOLOGY: A DEVELOPMENT METHODOLOGY

Olavo Mendes
DECOM/CCHLA/UFPB
Federal University at Paraiba – Brazil

PhD Student Cognitive Informatics
Quebec University at Montreal - UQAM
olavomendes@hotmail.com

Alain Abran
École de Technologie Supérieure - ETS
1100 Notre-Dame Ouest, H3C 1K3 Montréal Québec , Canada,
aabran@ele.etsmtl.ca

Keywords: *SWEBOK, Software Engineering Body of Knowledge, Ontology, Ontology development, Ontology methodologies, SWEBOK Ontology*

Introduction

According to Gruber's definition an ontology [1] is "a formal specification of a conceptualization". A conceptualisation being a simplified, abstract way of perceiving a segment of the world (a piece of reality), for which we agree to recognize the existence of a set of objects and their interrelations, as well as the terms we use to refer to them and their agreed meanings and properties.

Thus, ontologies represent a consensual, shared description of the pertinent objects considered as existing in a certain domain of knowledge (the domain of discourse). They constitute a special kind of software artefact conveying a certain view of the world (conceptualization), specifically designed with the purpose of explicitly expressing the intended meaning of a set of agreed existing objects.

Ontologies could play an important role in Software Engineering, as they do in other disciplines, where they: 1) provide a source of precisely defined terms that can be communicated across people, organisations and applications (information systems or intelligent agents); 2) offer a consensual shared understanding concerning the domain of discourse; 3) to render explicit all hidden assumptions concerning the objects pertaining to a certain domain of knowledge [2].

Despite some initial efforts to develop partial (sub domain) ontologies [3] [4] [5] [6], as a field of knowledge, Software Engineering still does not have a comprehensive detailed ontology which describes the concepts that domain experts agree upon, as well as their terms, definitions and meanings. Such ontology would also need to look at the more pertinent interrelations where concepts participate in the creation of the semantic network in which they are inserted.

The development of a "software engineering domain ontology" will allow us to: 1) share and reuse all knowledge accumulated until now in the Software Engineering field; 2) open news avenues to automatic interpretation *of this knowledge, using information systems or* intelligent *software agents.*

2 The SWEBOK Project

The SWEBOK project - Software Engineering Body of Knowledge [7] [8], is the result of a collaborative effort between the IEEE Computer Society and Université du Québec (École de Technologie Supérieure and UQAM). Over the years, close to 500 reviewers from very diverse domains including the industrial and academic fields, government agencies, professional societies, international standard organisation, as well as research centers, have been involved in the project, which has thus earned an international reputation in the software engineering field.

The resulting SWEBOK Guide is the result of great effort of declarative and procedural knowledge mining, acquisition and structuring that was, until then, scattered in an myriad of very diverse documents (scientific papers, congress proceedings, books, chapters, technical reports, technical standards), and of background knowledge from field experts, consultants and researchers.

The SWEBOK project team established the project with five objectives [7]:

1) To characterize the contents of the software engineering discipline; 2) To provide topical access to the software engineering body of knowledge; 3) To promote a consistent view of software engineering worldwide. 4) To clarify the place—and set the boundaries—of software engineering with respect to other disciplines such as computer science, project management, computer engineering, and mathematics; 5) To provide a foundation for curriculum development and individual certification material.

The SWEBOK project allowed to build a consensus (using the Delphi technique) on: 1) the knowledge areas consensually agreed to integrate the software engineering field; 2) the knowledge content associated to each domain, as well as the related major references; 3) the scientific disciplines participating in each area of knowledge.

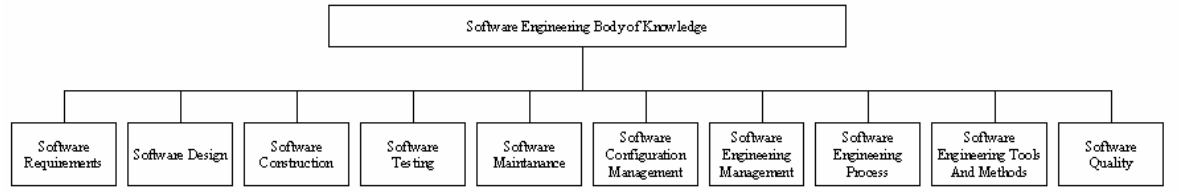


Figure 1: **Knowledge Areas of the Software Engineering Body of Knowledge [7] [8]**

The resulting product of the SWEBOK project it is not the body of knowledge itself, but rather a guide to it, permitting to gain consensus on the core subset of knowledge characterizing the software engineering discipline [7] [8]. As a result, ten knowledge areas have been identified as integrating the Software engineering field: KA.01 Software requirements, KA.02 Software design, KA.03 Software construction, KA.04 Software testing, KA.05 Software maintenance, KA.06 Software configuration management, KA.07 Software engineering management, KA.08 Software engineering process, KA.09 Software engineering tools and methods, KA.10 Software quality.

3 Project Goal

Our ultimate project goal is to build and validate an ontology for the Software engineering field, using the knowledge already acquired, structured, validated and made available, by the

SWEBOK project in the form of the SWEBOK Guide (last version Iron Man, 18.05.2004), as well as other scientific knowledge sources such as technical standards (ISO and IEEE).

Besides the benefits already mentioned in section 1, the use of the "software engineering ontology" which is a result of this project may also contribute to the development of additional content validation by automatic cross-correlation validation (besides that which is already done already done continuously by the SWEBOK review team) across the ten areas of knowledge integrated in the SWEBOK Guide. This would ensure that all concepts and definitions are used in a consistent fashion throughout all SWEBOK's areas of knowledge. An automatic validation would also be useful in the ISO/IEC JTC1/SC-7 SWG5 development toward's the harmonisation of all vocabulary used by the various working groups involved in software engineering technical standards.

4 The Problem

The ontology development process involves many activities that can present a high level of complexity, depending on the intended scope, size and level of detail of the ontology under construction [9] [10] [11].

As a consequence, the construction of an ontology cannot be conducted in an improvised or ad hoc fashion. The complexity of activities like conceptualisation, knowledge structuring/ontologisation, ontology evaluation, etc., require the use of management processes, in order to control cost, risks, schedules and to ensure that the artefacts produced are of the intended quality..

An important number of methodologies are presently described in the literature. The problem however is that 1) there is presently no consensus about the best practices to adopt concerning the construction of an ontology; 2) these ontology development methodologies make use of different construction methods, and frequently offer guidance to different portions of the ontology development cycle; 3) finally, until now, the ontology development process did not have any technical standard (official or *de facto*) to guide the development process, despite major efforts in this direction.

Thus a number of questions remain open:

- Which ontology development methodology provides the best guidance to attaint our established goal (the development of comprehensive software engineering ontology)?
- Which life-cycle model (cascade, incremental prototyping, evolutionary prototyping, etc.) is best suited to the planned ontology development?
- Which are the inputs, outputs and activities to be performed in order to develop the aimed ontology?
- Which are the key activities in the ontology development process?

This paper presents some preliminary results aimed at answering the above stated questions.

5 Methodology

In order to attain the stated goal, the following activities have been developed in this study:

- A detailed literature review of the ontology development methodologies;
- Preliminary classification of construction methodologies according to the mode of construction. Special emphasis was given to methodologies permitting ontology construction from scratch (Figure 2);
- Analysis of the ISO/IEC 12207-95 software life-cycle standard;

- Analysis of the surveyed methodologies from the ISO/IEC 12207-95 perspective;
- Preliminary identification of the differences and commonalities between the stated ontology development activities and the ISO/IEC standard;
- Proposal of a conceptual framework to compare and analyse the ontology development activities considered in the different methodologies;
- Additional literature reviews and refinement of the proposed conceptual framework;
- Identification and comparison of the ontology development activities proposed in the methodologies surveyed;
- Identification of the most and least frequently mentioned activities;
- Identification of key activities,

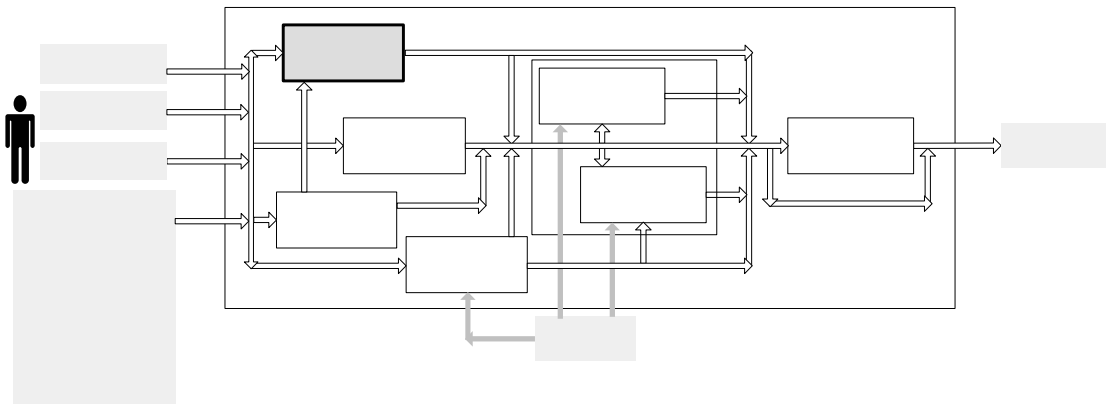


Figure 2: **Framework for ontology development methodologies**

6 Results

Some of the results produced by this study include:

A total of forty-eight ontology development methodologies have been identified in the literature review. Among these, fourteen corresponding to methodologies for the construction from scratch (the most recent in 2003) and seven for ontology evaluation. These figures indicate the dynamics in this area of research and the lack of an international standard or even of a *de facto* standard.

Leading ontology development methodologies authors [9] [11] [12] [13], agree that the process must be managed like any other software development project, in order to ensure that cost, schedule, risk and quality of the produced artefacts always remain under control. Nevertheless some project phases like Feasibility study, Project Planning, Tracking and Control are mostly absent from the methodologies surveyed. Configuration management, and quality assurance are also activities which are somehow absent. Despite being considered as primary life-cycle processes in the software development life-cycle, activities such as Deployment, Utilisation and Maintenance, are still very absent from the surveyed methodologies.

The activities mostly frequently mentioned in the literature are: Ontology specification, Conceptualisation, Ontologisation and Implementation. Authors consider these to be the three key ontology development activities.

It must be noted however, that there is a wide variation between methodologies concerning the terms used to name these activities, and the boundaries which define them. Sometimes, certain activities are absent or amalgamated with others.

Ontology evaluation and integration are examples of activities which share a large consensus between the surveyed methodologies that must be present in the process of ontology development.

Finally, among the fourteen ontology development methodologies surveyed, only two have a sufficient degree of coverage and detailed guidelines for users (domain experts and knowledge engineers/ontologists). We will adopt the guidance principles and activities prescribed by these methodologies in our project to develop a comprehensive Software Engineering ontology.

7 Towards a Software Engineering Ontology

We have chosen to implement the SWEBOK ontology using the OWL formalism due to its knowledge representation capabilities (by defining classes, individuals, properties, relationships in which these classes participates and axioms), and the possibility to reason about these classes and individuals. Other major web ontology languages are: SHOE (1996), XML (1996, 97), RDF (1997), OIL (late nineties), DAML – DARPA (2000), DAML+OIL (2001). OWL, the Web Ontology Language is the more recently ontology language (2001, Feb 2004).

At the root class of the ontology we find a concept, which corresponds to the SWEBOK Guide. Under this class (subclass of owl: Thing, a class that contains all classes), we find the main classes corresponding to the ten areas of knowledge that integrate the Guide, linked to the root class by the hasParts property. Each area of knowledge represents the agreed knowledge about the domain class, and can be successively exploded, revealing new classes with growing levels of detail. An example of the SWEBOK ontology (presented in the OWL formalism) is depicted at figure 3 (corresponding to the SWEBOK main level presented in figure 1).

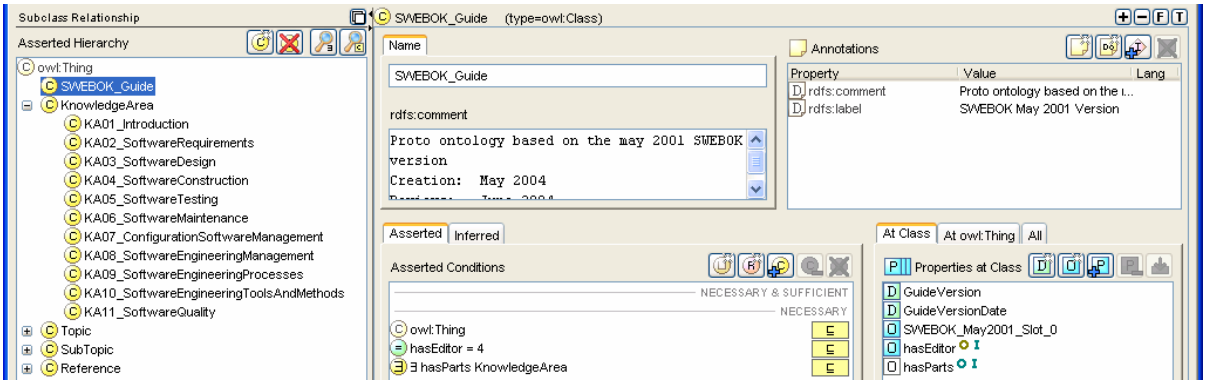
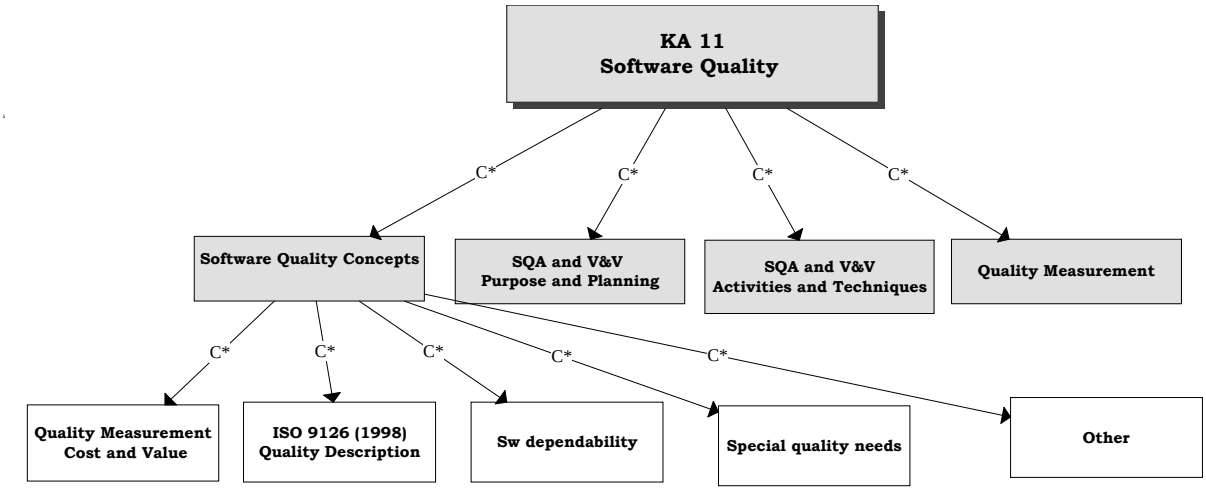


Figure 3: *The SWEBOK Ontology main level*

The classes (superclasses and subclasses) are organized in a structured hierarchy, using generalization/specialisation links to produce a taxonomy. Other types of links are also present (ex: contains, hasTopic, defines, and the inverse relations pertainsTo, isTopicOf, isDefinitionOf, etc.), capturing the existing semantics conveyed by multiples concept associations.

A zoom on the concept representing chapter 11 of the SWEBOK guide, reveals additional concepts, representing the knowledge associated with this topic. Figure 4 presents the four

subtopics which exist under the Software Quality topic. The C links represents the hasParts link with a* many *cardinality.*



The above topic of the SWEBOK ontology represented in the OWL formalism is depicted in figure 5. Topics contained in the area of Software Quality knowledge are shown in the left side panel. The subtopics integrating the first element (Software quality concepts) are also partially shown. The central widget (asserted conditions) are used to compose the axioms (logical expressions) that describes (using a set of necessary conditions) and define (using sets of necessary and sufficient conditions) the concepts that integrate the SWEBOK ontology.

With concept KA11 Software Quality *as an example, some axioms are shown: the KA11 Software Quality is an area of Knowledge, part of the SWEBOK guide that has other areas (mutually disjointed). The axioms describe also that Software Quality is composed of four topics (Quality concepts, SQA and V&V Purpose and planning and SQA and V&V Activities and techniques)*

The OWL widget (at the right side) contains the properties (attributes and relationships, describing the concept and linking this one to other concepts). As an example, three inherited and modified (locally overriden) properties are shown: KA11 has authors, is part of the SWEBOK guide and has topics (four already mentioned).

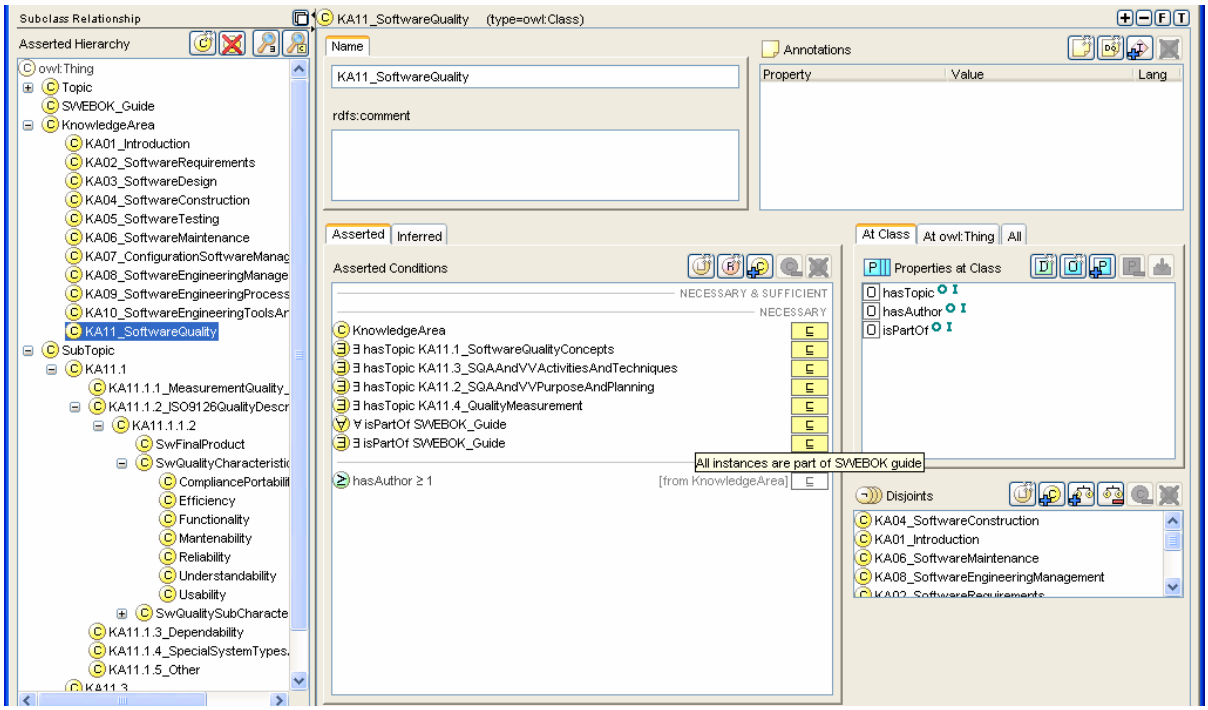


Figure 5: The Software Quality Ontology in OWL(a partial view), levels 1 and 2

8 Conclusion

This paper has presented the results of the first phase of a project aimed at developing a comprehensive ontology of the Software Engineering field. The major contributions provided by this study are: 1) Identification of the ontology development methodologies providing the best guidance to attaint our established goal; 2) Identification of a life-cycle model best suited to the planned ontology development; 3) Identification of main inputs, outputs and activities to be performed in order to develop the aimed ontology; 4) Identification of key activities in the ontology development process. Some preliminary results of the software quality ontology are also presented and developed using the May 2001 version of the SWEBOK Guide.

References

[1] T.R. Gruber. *Towards Principles for the Design of Ontologies Used for Knowledge Sharing*. In Roberto Poli Nicola Guarino, editor, International Workshop on Formal Ontology, Padova, Italy, 1993. Technical report KSL-93-04, Knowledge Systems Laboratory, Stanford University.

[2] Gruninger, M., Lee, Jintae., *Ontology Design and Applications*, Communications of the ACM, February 2002, 45 (2), 1-2, 2002.

[3] C. Wille, A. Abran, J-M Desharnais, R. Dumke, *The Quality concepts and sub concepts in SWEBOK: An ontology challenge*, in International Workshop on Software Measurement (IWSM) , Montreal , 2003 , pp. 18,

[4] C. Wille, R. Dumke, A. Abran, J-M, Desharnais, *E-learning Infrastructure for Software Engineering Education: Steps in Ontology Modeling for SWEBOK*, in Ontology Modeling for SWEBOK , in Software Measurement European Forum , Rome, Italy, 2004

[5] A. Qasem, *A prototype DAML+OIL Ontology IDE*, International Semantic Web Working Symposium, Stanford, 2001. http://www.semanticweb.org/SWWS/program/position/soi-qasem.pdf

[6] A. Qasem, *The WOSE Portal*, http://java-emporium.com/projects/wose/index.html

[7] P. Bourque, R.L. Dupuis, A. Abran, T*he Guide to the Software Engineering Body of Knowledge*, IEEE Software, November/December, 1999.

[8] A. Abran, J. Moore, P. Bourque, R.L. Dupuis, L. Tripp, *Guide to the Software Engineering Body of Knowledge – SWEBOK*, Trial Version 1.0, IEEE-Computer Society Press, May 2001, URL: http://www.swebok.org

[9] M. Uschold and Michael Gruninger, *Ontologies; principles, Methods and Applications*, Knowledge Engineering Review, Vol 11, No 2, Jun 1996

[10] D. Jones, T. Bench-Capon and P. Visser, *Methodologies for Ontology Development*, Proceedings of the IJCAI-99 workshop on Ontologies and Problem-Solving Methods, 1999

[11] R. Mizoguchi, *Fundamental Aspects of Ontology Engineering*, to appear in Proceedings of the ACFAS Congress, Colloque d'Informatique Cognitive (C622), Montréal, May 2004

[12] M. Uschold, and M. King, *Towards a Methodology for Building Ontologies*. Proceedings of IJCAI95's Workshop on Basic Ontological Issues in Knowledge Sharing. 1995.

[13] M. Fernandez, A. Gomez-Perez, and N. Juristo, *METHONTOLOGY: From Ontological Art to Ontological Engineering*. In Workshop on Knowledge Engineering: Spring Symposium Series

Ebert, C.; Dumke, R.; Bundschuh, M.; Schmietendorf, A.:

Best Practices in Software Measurement

Springer Publ., 2004 (320 pages)
ISBN 3-540-20867-4

The software business is challenging enough without having to contend with recurring errors. One way repeating errors can be avoided is through effective software measurement. In this book is offered a practical guidance built upon insight and experience. The authors detail knowledge and experiences about software measurement in an easily understood, hands-on presentation and explain many current ISO standards.

Dumke, R.; Abran, A.: (Eds.):

Investigations in Software Measurement

Shaker Publ., Aachen, 2003 (326 pages)
ISBN 3-8322-1880-7

The book includes the proceedings of the 13th International Workshop on Software Measurement (IWSM2003) held in Montreal in September, 2003, which constitute a collection of theoretical studies in the field of software measurement and case reports on the application of software metrics in companies and universities in Argentinia, Canada, Finland, Germany, India, Italy, Japan and the Netherlands.

John C. Munson, PH.D.:

Software Engineering Measurement

Auerbach Publications, Boca Raton, Florida, 2003 (443 pages)
ISBN 0-8493-1503-4

The author describes how to manage software development measurement systems, how to build software measurement tools and standards, and how to construct controlled experiments using standardized measurement tools.

The book answers three fundamental questions. First, exactly how do you get the measurement data? Second, how do you convert the data from the measurement process to information that you can use to manage the software development process? Third, how do you manage all of the data?

By demonstrating how to develop simple experiments for the empirical validation of theoretical research and showing how to convert measurement data into meaningful and valuable information, Software Engineering Measurement will show you how to use your measurement information for immediate, software process improvement.

Endres, A.; Rombach, D.:

A Handbook of Software and Systems Engineering

Pearson Education Limited, Essex, 2003 (327 pages)
ISBN 0-321-15420-7

Computers are the most pervasive tools of modern society. Their development relies on advanced methods of software and systems engineering. Based on repeated and consistent observations, key lessons of these fields can now be formulated into rules or even laws, providing initial building blocks towards a theoretical foundation that is essential for further research, for teaching and for the practice of software development.

Intended as a handbook for students and professionals alike, this book is the first to identify and discuss such rules and laws. They are largely independent of technologies, and thus form a basis for the principles underlying software and system engineering. Software and system engineers should be aware of this proven body of knowledge, to ensure professionalism and due diligence in their work.

The book is structured around the software development lifecycle. It begins with requirements definition and goes on to maintenance and withdrawal. In different process models, these tasks have different importance or are applied in a different sequence, or even cyclically.

Büren, G.; Bundschuh, M.; Dumke, R.: (Eds.):

Software-Messung in der Praxis

Shaker Publ., Aachen, 2003 (169 pages)
ISBN 3-8322-2146-8

The book includes the proceedings of the DASMA Metric Conference **MetriKon2003** held in Ulm in November, 2003, 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.

Pandian, C. R.:

Software Metrics – A Guide to Planning, Analysis, and Application

CRC Press Company, Boca Raton, 2004 (286 pages)
ISBN 0-8493-1661-8

The book simplifies software measurement und explains its value as a pragmatic tool for management. Ideas and techniques presented in this book are derived from best practices. Some of the keywords are fundamentals of software measurement, metrics system architectures, regression models, exploring metrics for defect management, and strategic visions.

PE2004:

5. Workshop Software Performance Engineering
14. Mai 2004 in München,
see: `http://ivs.cs.uni-magdeburg.de/~schmiete/peak/`

ISESE 2004:

IEEE International Symposium on Empirical Software Engineering
August 19-20, 2003, Redondo Beach, CA
see: `http://www.cs.umd.edu/~mvz/isese2004/`

Metrics 2004:

10th International Symposium on Software Metrics
September 14-16, 2004, Chicago
see: `http://swmetrics.org/`

ASQT 2004:

Softwarequalität und Test 2004
September 15-17, 2004, Klagenfurt, Austria
see: `http://www.asqt.org/`

IFPUG 2004:

IFPUG 2004 Annual Conference
September 19-24 , 2004, San Diego
see: `http://www.ifpug.org/conferences/`

CONQUEST 2004:

Conference on Quality Engineering in Software Technology
September 22-24, 2004, Nuremberg, Germany
see: `http://www.conquest 2004.de`

UML 2004:

Fourth International Conference on the Unified Modelling Language
October 11-15, 2004, Lisbon, Portugal
see: `http://www.umlconference.org/`

IWSM2004/Metrikon2004:

14th International Workshop on Software Measurement, DASMA Metrik Kongress
November 3-5, in Berlin, Germany
see: `http://iwsm2004.cs.uni-magdeburg.de`

EuroSPI 2004:

European Conference on Software Process Improvement
November 10 - 12, 2004, Trondheim, Norway
see: `http://www.eurospi.net/`

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

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

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

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

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

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

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

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

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

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

- **http://www.utexas.edu/coe/sqi/**
  Software Quality Institute of the University of Texas at Austin. Offers comprehensive general information sources on software quality issues.

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

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

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

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

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

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

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

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

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

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

- **http://www.iese.fhg.de/**
  Homepage of the Fraunhofer Institute for Experimental Software Engineering (IESE) in Kaiserslautern, Germany

- **http://www.highq.be/quality/besma.htm**
  Homepage of the Belgian Software Metrics Association (BeSMA) in Keebergen, Belgium

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

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

News Groups

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

Software Measurement Associations

- **http://www.aemes.fi.upm.es**
  AEMES Association Espanola de Metricas del Software
- **http://www.asqf.de**
  ASQF Arbeitskreis Software-Qualität Franken e.V., Nuremberg, Germany
- **http://www.cosmicon.com**
  COSMIC Common Software Measurement International Consortium
- **http://www.dasma.org**
  DASMA Deutsche Anwendergruppe für SW Metrik und Aufwands-schätzung e.V.
- **http://www.esi.es**
  ESI European Software Engineering Institute in Bilbao, Spain
- **http://www.mai-net.org/**
  Network (MAIN) Metrics Associations International
- **http://www.sttf.fi**
  FiSMA Finnish Software Metrics Association
- **http://www.iese.fhg.de**
  IESE Fraunhofer Einrichtung für Experimentelles Software Engineering
- **http://www.isbsg.org.au**
  ISBSG International Software Benchmarking Standards Group, Australia
- **http://www.nesma.nl**
  NESMA Netherlands Software Metrics Association
- **http://www.sei.cmu.edu/**
  SEI Software Engineering Institute Pittsburgh
- **http://www.spr.com/**
  SPR Software Productivity Research by Capers Jones
- **http://fdd.gsfc.nasa.gov/seltext.html**
  SEL Software Engineering Laboratory - NASA-Homepage
- **http://www.vrz.net/stev**

STEV Vereinigung für Software-Qualitätsmanagement Österreichs

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

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

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

Software Metrics Tools (Overviews and Vendors)

Tool Listings

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

- **http://mdmetric.com/meastl1.htm**
  Maryland Metrics Tools

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

Tool Vendors

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

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

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

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

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

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

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

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

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