Reiner R. Dumke René Braungarten
Günter Büren Alain Abran
Juan J. Cuadrado-Gallego (Eds.)

Software Process and Product Measurement

LNCS 5338

International Conferences
IWSM 2008, MetriKon 2008, and Mensura 2008
Munich, Germany, November 2008, Proceedings

Springer

Lecture Notes in Computer Science 5338

Commenced Publication in 1973
Founding and Former Series Editors:
Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

Reiner R. Dumke René Braungarten
Günter Büren Alain Abran
Juan J. Cuadrado-Gallego (Eds.)

Software Process and Product Measurement

International Conferences
IWSM 2008, MetriKon 2008, and Mensura 2008
Munich, Germany, November 18-19, 2008
Proceedings

Springer

Volume Editors

Reiner R. Dumke
Otto-von-Guericke-Universität Magdeburg
Institut für Verteilte Systeme
Magdeburg, Germany
E-mail: dumke@ivs.cs.uni-magdeburg.de

René Braungarten
Otto-von-Guericke-Universität Magdeburg
Institut für Verteilte Systeme
Magdeburg, Germany
E-mail: braungar@ivs.cs.uni-magdeburg.de

Günter Büren
Büren & Partner Software-Design GbR
Nürnberg, Germany
E-mail: gb@bup-nbg.de

Alain Abran
École de technologie supérieure
Département de génie logiciel et des TI
Montréal, Québec, Canada
E-mail: alain.abran@etsmtl.ca

Juan J. Cuadrado-Gallego
Universidad de Alcalá
Edificio Politécnico
Alcalá de Henares, Madrid, Spain
E-mail: jjcg@uah.es

Library of Congress Control Number: 2008939385

CR Subject Classification (1998): D.2.8, D.2, K.6.1-4, J.1

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743
ISBN-10 3-540-89402-0 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-89402-5 Springer Berlin Heidelberg New York

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India
Printed on acid-free paper SPIN: 12566945 06/3180 5 4 3 2 1 0

Preface

Since 1990 the International Workshop on Software Measurement (*IWSM*) has been celebrated annually alternating between Montréal (Canada) and various cities across Germany. The Montréal editions have been organized by the Software Engineering Research Laboratory (GELOG)[1] of the École de technologie supérieure - Université du Québec, which is directed by Prof. Alain Abran. The German editions have been organized jointly by the Software Measurement Laboratory (SMLAB)[2] of the Otto von Guericke University Magdeburg (Germany), which is directed by Prof. Reiner R. Dumke; and the German association for software metrics and effort estimation (DASMA e. V.)[3], which is led by Manfred Bundschuh and Günter Büren. The biennial editions of IWSM in Germany has been held jointly with the DASMA Software Metrics Congress (*MetriKon*)[4] since 2002. MetriKon is a yearly event, conducted every other year for a German-speaking audience at changing national locations for best-practice sharing of software measurement topics, bringing the best and renowned German-speaking experts of the field together.

The first two editions of the International Conference on Software Process and Product Measurement (*Mensura*) were organized by Juan J. Cuadrado-Gallego[5] from the University of Alcalá (Spain) and convened in Cádiz (Spain) in 2006 together with IWSM in Palma de Mallorca (Spain) in 2007. To foster research, practice and exchange of experiences and best practices in software processes and product measurement, the 2008 editions of IWSM / MetriKon / Mensura were combined. The conferences were held during November 18–20, 2008 in Munich (Germany) and kindly hosted by Siemens AG.

This volume comprises the proceedings of IWSM / MetriKon / Mensura 2008 and consists of the final papers presented at these joint events. Each one of these papers has been thoroughly revised and extended in order to be accepted for publication. The IWSM / MetriKon / Mensura Steering Committee is proud to have–once more–obtained the approval of Springer to publish the second edition of the joint conference proceedings in the prestigious *Lecture Notes in Computer Science* (LNCS) series. We hope to maintain this collaboration for the future editions of these joint events.

November 2008

Reiner R. Dumke
René Braungarten
Günter Büren
Alain Abran
Juan J. Cuadrado-Gallego

[1] http://www.lrgl.uqam.ca/
[2] http://ivs.cs.uni-magdeburg.de/sw-eng/us/
[3] http://www.dasma.org/
[4] http://www.metrikon.org/
[5] http://www.cc.uah.es/jjcg/

Organization

General Chairs

Manfred Bundschuh	DASMA e. V., Germany
Reiner R. Dumke	Otto von Guericke University, Magdeburg, Germany
Alain Abran	University of Québec / ÉTS, Montréal (Québec), Canada
Juan J. Cuadrado-Gallego	University of Alcalá, Madrid, Spain

Organization Chair

Günter Büren	Büren & Partner, Nuremberg, Germany

Proceedings Chair

René Braungarten	Bosch Rexroth AG, Lohr am Main, Germany

Program Committee Chair

Reiner R. Dumke	Otto von Guericke University, Magdeburg, Germany

Program Committee

Luigi Buglione	Engineering.IT S.p.A., Italy
François Coallier	ÉTS, Montréal (Québec), Canada
Ton Dekkers	Galorath International Ltd., UK
Jean-Marc Desharnais	ÉTS, Montréal (Québec), Canada
José Javier Dolado	University of the Basque Country, San Sebastian, Spain
Axel Dold	Daimler AG, Sindelfingen, Germany
Christof Ebert	Vector Consulting, Stuttgart, Germany
Bernd Gebhard	BMW AG, Munich, Germany
Marcela Genero	University of Castilla-La Mancha, Ciudad Real, Spain
Naji Habra	FUNDP, Namur, Belgium
Nadine Hanebutte	University of Idaho, Moscow (Idaho), USA
Hans-Georg Hopf	GSO-Hochschule, Nuremberg, Germany

Claus Lewerentz	Technical University Cottbus, Cottbus, Germany
Marek Leszak	Alcatel-Lucent, Nuremberg, Germany
Peter Liggesmeyer	Fraunhofer IESE, Kaiserslautern, Germany
Mathias Lother	Robert Bosch GmbH, Stuttgart, Germany
Fernando Machado	Catholic University of Uruguay, Montevideo, Uruguay
Roberto Meli	DPO, Rome, Italy
Dirk Meyerhoff	Schueco-Service GmbH, Bielefeld, Germany
Jürgen Münch	Fraunhofer IESE, Kaiserslautern, Germany
Olga Ormandjieva	Concordia University, Montréal (Québec), Canada
Frances Paulisch	Siemens AG, Munich, Germany
Ricardo J. Rejas-Muslera	University Francisco de Vitoria, Madrid, Spain
Salvador Sánchez-Alonso	University of Alcala, Madrid, Spain
Andreas Schmietendorf	Berlin School of Economics, Germany
Harry Sneed	SES, Munich/Budapest, Germany/Hungary
Charles Symons	Software Measurement Service Ltd., Edenbridge, UK
Hannu Toivonen	Nokia Siemens Networks, Finland
Cornelius Wille	University of Applied Sciences, Bingen, Germany
Loreto Zornoza	IBM, Spain
Horst Zuse	Technical University Berlin, Berlin, Germany

Tutorial Chairs

Ralf Russ	Siemens AG, Munich, Germany
Günter Büren	Büren & Partner, Nuremberg, Germany
Marek Leszak	Alcatel-Lucent, Nuremberg, Germany

Conference Support

Romy Gampe	DASMA e. V., Germany
Dagmar Dörge	Otto von Guericke University, Magdeburg, Germany
Carsten Peitscher	Signal-Iduna, Dortmund, Germany
Helmut Benesch	Siemens AG, Munich, Germany

Sponsors

We wish to express our gratitude to the sponsors of the IWSM / MetriKon / Mensura 2008 for their essential contribution to the conference.

Organizers

Moreover, we also wish to express our gratitude to the organizers of IWSM / MetriKon / Mensura 2008 for their tireless dedication:

Table of Contents

Session A5 – Product Measurement

Session B5 – Measurement Methodology II

Project Sizing and Estimating: A Case Study Using PSU, IFPUG and COSMIC

Luigi Buglione[1], Juan J. Cuadrado-Gallego[2], and J. Antonio Gutiérrez de Mesa[2]

[1] École de Technologie Supérieure (ÉTS) / Engineering.it
luigi.buglione@eng.it
[2] Universidad de Alcalá. Edificio Politécnico. Autovía A2, Km. 31,
7. 28805 - Alcalá de Henares, Madrid, Spain
{jjcg,jagutierrez}@uah.es

Abstract. From the late '70s on, Albrecht's Function Point Analysis provided an insightful way to size a software system moving from the elicitation of Functional User Requirement (FUR), making an evaluation more objective than done before using Lines of Code (LOC). This technique has currently a plenty of variants, some of them become international de jure standards (e.g. COSMIC, NESMA, Mark-II and FISMA) - called FSM (Functional Size Measurement) methods - and they are widely adopted worldwide. A common problem when using a FSM for estimation purposes is that the software size (that is a product measure, referring only to its functional side) is used as the solely independent variable to estimate the overall project effort, that includes the effort of both the functional and non-functional activities within the project's boundary, as currently stressed more and more in the Scope Management field, also in the Software Engineering domain (see NorthernScope and SouthernScope approaches), not knowing neither the approximated distribution between the two parts. This missing information, usually not gathered in projects' repositories, can be one of the reasons leading to a lower capability in estimating project effort.

In 2003, a new technique called PSU (Project Size Unit) come out with the aim to size the 'project' entity from a Project Management viewpoint. It can be used alone or jointly with a FSM unit. In the second case, the joint usage of the two values can improve what a FSM cannot measure and therefore estimate, that is the non-functional side of a software project. This paper presents a case study with 33 projects measured both with IFPUG FPA and COSMIC methods as well as with PSU, showing the obtained results using the different sizes for estimating the overall effort, and providing a rationale for the better results with PSU.

Keywords: Estimation, Function Points, Project Size Unit (PSU), Case Study, Non-Functional Requirements, Scope Management.

1 Introduction

When dealing with every activity in the real world, a common strategy is firstly to apply a top-down view on the entity of interest and then to refine and integrate

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 1–16, 2008.

information with a bottom-up view. Shifting this concept to the estimation process, we need before to shape the logical boundary for the activity to perform, in order to properly understand – approximately - the amount of resources needed and consequently the time and costs such activity will require.

But when a software project must be analyzed in the feasibility phased and then planned, the above described approach often seems to be difficult to be applied. Observing the experiences in ICT companies as well as reading them in technical papers, it seems there is a large distance between the experiential estimations and a statistical usage of its own project data. And there is a tendency to use very few numbers – typically product measures - in order to estimate time and costs for the overall project.

During last years the "*scope management*" approach from the Project Management domain [1] come in also in the Software Engineering one: some examples are the SouthernScope [2] and the NorthernScope [3] approaches, integrating the usage of functional size measurement methods with other values and thoughts able to properly represent the whole project' scope. Again, another technique called Project Size Unit (PSU) was created in 2003 for trying to catch the overall project size and some experiences have been done with it [4].

The objective of this paper is to describe the PSU technique and discussing the way it can be used with or without a FSMM for refining project's estimations, taking always in mind that the final goal is to achieve improvements in estimating projects, and that size units – whatever they are – are the way to reach that goal, not the goal itself.

Section 2 discusses the estimation issue using a FSM method, delimiting the scope and boundary for such methods. Section 3 presents the basics for PSU and the way it can be also used jointly with a FSMM. Section 4 presents a case study with the analysis of 33 sample projects sized with IFPUG v4.2 [5], COSMIC v2.2 [6] and PSU v1.01 [7], proposing first results and thoughts for improving project estimations. Section 5 will conclude with a summary of what discussed and next work planned on this issue.

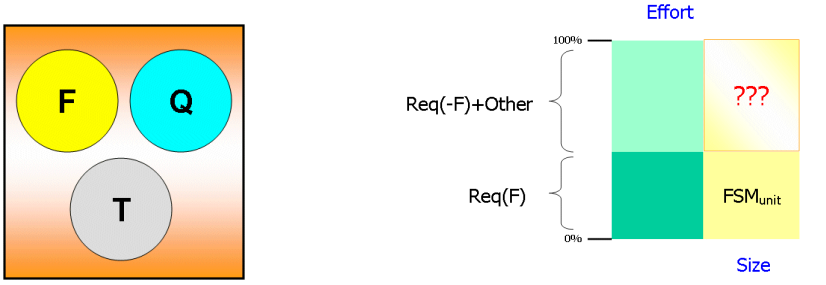
2 FSM and Estimation

2.1 What a FSM Method Size (And What Not)

According to the ISO/IEC 14143-1 standard [8][9], a functional size measurement method (FSMM) takes into account only the so-called FUR (Functional User Requirements), discarding the other ones – explicit and implicit ones – called in the latest version simply "non-functional requirements"[1]. Figure 1 shows the 1998 (software) product requirement classification into F/Q/T types and the relationships between Effort and Size against the project requirement types.

The direct consequences from this ISO clarification was the exclusion of the adjustment factors in the FSMM methods standardized from the final value (i.e. the ISO/IEC 20926:2003 for IFPUG CPM v4.1 considers only the first five steps in the calculation process, calculating the solely UFP value). The rationale is that the

[1] The 1998 version [8] split the non-functional part into Quality and Technical Requirements. This requirement classification for a software product (F/Q/T) was also received by IFPUG [5].



(a) (b)

Fig. 1. (a) Requirement types according to ISO 14143-1:1998; (b) Relationship between Effort and Size against project requirements (F vs NF)

non-functional side – as initially stated also in first's Albrecht's 1979 paper on FPA [10] – has to be treated separately but in a parallel manner with the functional one. From a mathematical viewpoint, using the non-functional factors as adjustments produces effort under-estimation for such kind of tasks. A simple example can be in IFPUG FPA a TDI value lower than 35 points (therefore a VAF lower than 1): the result would be a negative contribution on the unadjusted functional size, with a lower estimated number of man-days, even if a certain amount of man-days for non-functional activities would be anyway yet spent/planned. Again, from an economical viewpoint, it means that the cost/day of a role typically playing a non-functional job would be lower than those ones playing functional tasks. And it seems to do not properly shape what happens [11].

2.2 Estimation by a Functional Size Unit (fsu) with Some Open Questions

When dealing with whatever functional size unit (fsu), the typical way to estimate the project effort can be derived from:

- a regression equation (i.e. a linear one) based on its own data;
- productivity figures typical from a certain system (i.e. filtering by application type, development type, size range and technology used), according to its own data or from external sources (i.e. ISBSG repository);
- The crossing between the two above information.

Thus, there are some basic and open questions to be answered:

- Productivity, as currently defined and applied, is given by the ratio between the number of fsu and the overall project effort. It can be defined a 'nominal' productivity. Being the upper value referable to a product (and only for its functional portion), while the lower value refers to the overall project (including therefore the effort for all the types of requirements: F/Q/T/O), is it a valuable number to consider for deriving projects estimates?
- Since a fsu is a valid measure only for the functional part of a software product, what about its non-functional part?

3 PSU: Project Size Unit

3.1 Background

In 2003, during the path towards a Sw-CMM [12] ML3 certification process in an organizational unit (OU) of c.a. 80 people from a large ICT multinational company, one of the first questions to solve was to accomplish requirements from the Software Project Planning key process area, requesting to estimate efforts and costs (PP, Ac10), taking care of the overall project scope (PP, Ac2)[2].

Since the projects managed by such OU were typically TLC and Energy/Utility projects with an average 55-65% functional effort, with no enough time to properly train people with a FSMM, the point was to find out another solution for achieving the final goal taking into account also those constraints, but not too revolutionary to require too much extra time to be learned and used.

3.2 Rationale

The idea was to move from the boundary of the activities planned and run within a project, using the same approach Albrecht adopted for FPA, but extending the scope to all the user requirements (UR) a project has, not only FUR (Functional User Requirements), but also the Non-Functional (NFR) ones. From a Project Management viewpoint it means to consider the whole amount of activities included in a WBS, trying to estimate such amount of effort from requirements in an early stage, referring to the ISO 9000 quality definition [13], that includes both explicit and implicit requirements, where both ones generate activities and therefore effort to be estimated and planned within the project boundary.

Looking at Figure 2, our goal was to find out a new measure at the project level for approximating in early stages the overall "project size" and obtain acceptable estimates overcoming the inner scope of a FSMM, that's a functional product size measure. 'Project Size' is a term not yet defined in the ISO/IEEE/PMI glossaries. Our

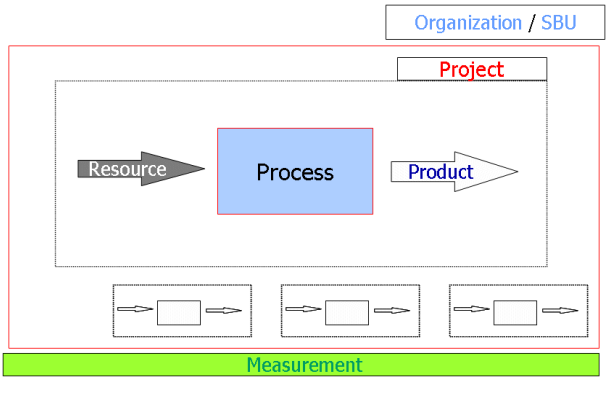


Fig. 2. STAR Taxonomy: measurable entities [14]

[2] The same happens also with the newer CMMI-DEV v1.2 [15] model, where the old SPP key process area was simply renamed Project Planning (PP).

proposal [11], according to the above premise, is to define it as "*the size of a software project, derived by quantifying the (implicit/explicit) user requirements referable to the scope of the project itself*". This term (and our own definition) was proposed to ISBSG for inclusion in a next revision of its Glossary of Terms [16].

Another objective to accomplish was to derive a mechanism valid for internal improvement first, and for external benchmarking in a second moment. The name for this new technique was **P**roject **S**ize **U**nit (**PSU**), definable as project management 'virtual' size technique.

3.3 Calculation Rules

Moving from the above premises, the FPA calculation rule was adapted to a project management logic. UFP are given by the sum of the 5 BFC (Base Functional Components) weighted by complexity.

In PSU the BFC corresponds to the WBS project tasks, firstly classified by nature: Management (M), Quality (Q) and Technical (T). The T-tasks refer to the primary processes, while the M/Q-tasks to the organizational and support processes. Other possible classifications of tasks are by requirement type (functional vs non-functional) and by SLC phase. All these classifications allow to easily gathering from early stages its own project historical data, which represent a foundation for PSU but for any process improvement initiative in general.

As in any good project management guideline, an activity should be always under control. The complexity of tasks is due by the effort of a task. The larger the effort for a task without any control/milestone in the middle, the more complex it is, therefore more risky and with higher probability to request a re-plan during the project lifetime. The PSU formula can be summarized as:

$$PSU = \sum_{i=M,Q,T} \sum_{j=H,M,L} task_i * weight_j \quad (1)$$

where the weights ranges can vary according to the organizational style and definition for creating projects' WBS and can be easily derived applying on a regular basis Pareto Analysis on the project historical database (PHD). Please refer to the PSU Measurement Manual (MM) for detailed procedures and tips [4].

Another PSU characteristic is to be *general-purpose*: because the BFC are tasks from a project, it has no limitation about application domain, as FSMM. Therefore it can be used for a whatever kind of project (i.e. service, building, performing arts, ..).

3.4 Automating PSU

Since the calculation rule simply counts tasks weighted by effort ranges, differently from a FSMM, PSU can be easily automated from a project WBS within a spreadsheet or – with a macro – directly in any PM tool, needing the time for a 'click' just when creating/modifying your project plan. Requirements for automation are available and an implementation under open source software (GanttProject[3]) was yet done [17]. The added value of an integration of PSU calculation within a PM tool is the possibility to export project's data (i.e. in xml) for an easier creation/update of the

[3] URL: www.ganttproject.org

organizational PHD, allowing several views on project's data as a base for next estimations [18][19].

3.5 PSU: When Calculate Them?

As suggested for FSMM, there are three typical moments in time for calculating it and gather values in the PHD: Feasibility study, Design phase and at the Closure phase.

3.6 PSU and FSM Methods

PSU is definable as a 'virtual' size measure because, differently from a FSMM, it needs an experiential/analogous estimate to produce a more refined estimate, compared with the 'organizational memory' (the PHD). Since the reduced time to calculate PSU, it can be used easily by SMEs what could not have time or resources for learning and applying a FSMM.

But it is possible also to use jointly PSU and FSMM: the advantage could be in early estimating the whole project effort with PSU with a better approximation than an early FSM method and after to fully calculate (also for contractual quests) fsu at the end of the Design and Closure phases.

3.7 PSU: Internal vs. External Comparability

IFPUG FPA allows an external comparability among projects worldwide because the system of weights and BFC ranges is the same from 1984 and never more modified. PSU born firstly as a technique for internal improvement, therefore changing periodically weights and effort ranges according to the closed projects entering into the PHD and reshaping the regression equations based on the updated database. In order to use PSU for external comparability, it is sufficient to make stable weights and effort ranges during time and/or among interested stakeholders [20].

3.8 PSU: Available Assets

All the PSU assets are freely available on the SEMQ website[4] in several languages[5]. Nowadays the downloadable assets are:

- Measurement Manual [4];
- MS-Excel calculation sheet (traditional / agile projects);
- Requirements for automating PSU [19].

4 A Case Study

4.1 Background and Objectives

During a B.Sc. 2006-07 Software Engineering course at the University of Alcalà de Henares (UAH, Spain), some students worked on learning and applying FSM methods such as IFPUG and COSMIC methodologies. Moving from a previous B.Sc.

[4] PSU webpage: www.geocities.com/lbu_measure/psu/psu.htm
[5] English, Spanish, Italian.

study about the conversion between IFPUG v4.2 and COSMIC v2.2 fsu, where 33 medium-sized projects were measured using both FSM methods [21] with a verification of the FSM count by an experienced senior measurer, the same projects were also sized with PSU v1.01 counting rules [22] [23] and some of the research questions above posed was investigated, in particular:

a) the relationship between PSU and IFPUG/COSMIC (if any);
b) which size unit among the three seems to be the better one for such dataset;
c) and of course, why.

4.2 Presentation of Data Sample

The basic data from the 33 sample projects are listed with details in the Annexes at the end of the paper. Some highlights (see Annexes B and C with full details):

- **Application type:** Management (16 projects), Management & Communication (6 projects), Management & Control (7), Management, Communication & Control (2), Application (2);
- **Estimated effort ranges:** From 493 up to 2589 man/days, with an average and median distribution by requirement type closely to 44-56% (F vs. NF). The classification of effort by SLC phase was done using the Spanish Government standard METRICA3 [24].

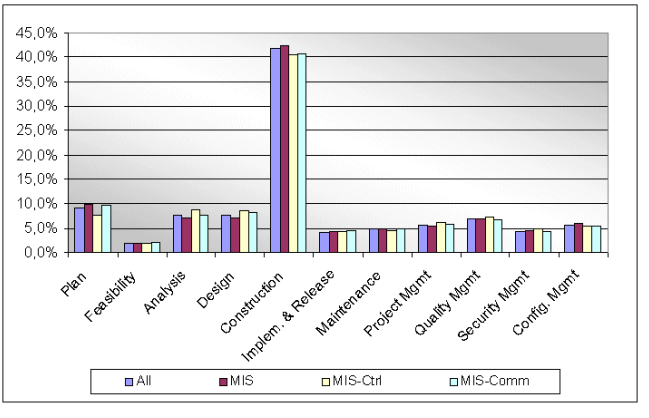


Fig. 3. Effort distribution by SLC phase according to METRICA3 [24]

Some highlights about the sizing measures (see Annexes B and C with full details):

- **Functional Size ranges:** From 109 up to 534 IFPUG UFP; from 41 up to 396 cfsu;
- **PSU weighting system:** The following values where assumed for the PSU calculation on the projects' sample:
  - Effort: three levels of complexity → High (26+ m/d), Medium (11-25 m/d), Low (0-10m/d);
  - Weights: H(1.8), M(1.4), L(1.0), that's an initial set of weights we experimented on such sample.

4.3 First Results

Linear regression analysis was performed using the three size units (taking care of their inner differences) in different combinations for building a size unit vs effort (using both the whole dataset and then by application type) estimation model. Since PSU values are the sum of two partial ones, derived from functional (PSU_f) and non-functional (PSU_{nf}) tasks elaboration, also PSU_f size was considered for being compared with IFPUG and COSMIC methods. About the first issue (effort estimation models), Table 1 summarizes the main results obtained (we discarded, obviously, those categories with too less projects):

Table 1. Some Estimation Models derived from the data sample

Id.	Relationship	Formula	R^2	Interpret.
Application Type: All; n=33				
1	PSU vs Effort	Y=4.4988x+183.23	0.5944	😐
2	PSU_f vs Effort	Y=5.1825x+669.97	0.2489	☹
3	UFP vs Effort	Y=-0.2767+1284.3	0.0019	☹
4	Cfsu vs Effort	Y=0.9057x+984	0.030	☹
Application Type: MIS; n=16				
5	PSU vs Effort	Y=5.2508x+145.3	0.7174	☺
6	PSU_f vs Effort	Y=5.5899x+781.62	0.2419	☹
7	UFP vs Effort	Y=-4.4025x+2738.1	0.1317	☹
8	Cfsu vs Effort	Y=0.6503x+1168+5	0.0072	☹
Application Type: MIS & Control; n=7				
9	PSU vs Effort	Y=3.6924x+208.04	0.6114	☺
10	PSU_f vs Effort	Y=5.3581x+500.7	0.4203	😐
11	UFP vs Effort	Y=7.2912x+1274.4	0.4068	😐
12	Cfsu vs Effort	Y=2.2822x+477.99	0.1912	☹
Application Type: MIS & Communication; n=6				
13	PSU vs Effort	Y=6.2849x+197.7	0.7552	☺
14	PSU_f vs Effort	Y=9.3033x+196.07	0.4332	😐
15	UFP vs Effort	Y=1.1943x+686.12	0.1351	☹
16	Cfsu vs Effort	Y=0.594x+917.38	0.0393	☹

From the observation of Table 1 results, it can be noted that in all cases PSU has a higher correlation with estimated effort than the other fsu, both IFPUG and COSMIC. This can be interpreted as a clear sign that there are some issues in projects that during the estimation phase having an influence on correlation; in particular:

- The non-functional effort (see the higher R^2 values for "PSU vs. effort" cases against the "PSU_f vs. effort" ones);
- A typical fsu is a *product*-level measure, therefore not covering such requirements, tasks and effort related to the *project*-level.

4.4 Applying PSU v1.21: A What-If Analysis

From the time of the comparative analysis, PSU calculation rules were modified. Instead taking into account M/Q tasks as an adjustment for T tasks (as well as VAF did referring to UFP), now all tasks – whatever their nature – are weighted by effort range. The difference comparing the same 33 sample projects sized with PSU v1.01

Table 2. Some Estimation Models derived from the data sample (PSU v1.21)

Id.	Relationship	Formula	R^2	Diff. %	Trend
Application Type: All; n=33					
1	PSU vs Effort	Y=4.2854x+32.067	0.6665	7.21	↑
2	PSU_f vs Effort	Y=5.2603x+222.13	0.6194	37.05	↑
3	UFP vs Effort	Y=-0.2767x+1284.3	0.0019	--	--
4	Cfsu vs Effort	Y=0.9057x+984	0.03	--	--
Application Type: MIS; n=16					
5	PSU vs Effort	Y=5.0612x-65.312	0.7844	0.70	↑
6	PSU_f vs Effort	Y=6.1357x+184.12	0.7129	47.10	↑
7	UFP vs Effort	Y=-4.4025x+2738.1	0.1317	--	--
8	Cfsu vs Effort	Y=0.6503x+1168.5	0.0072	--	--
Application Type: MIS & Control; n=7					
9	PSU vs Effort	Y=3.3145x+139.5	0.6800	6.86	↑
10	PSU_f vs Effort	Y=-1.0351x+1140.1	0.0802	-34.01	↓
11	UFP vs Effort	Y=7.2912x-1274.4	0.4068	--	--
12	Cfsu vs Effort	Y=2.2822x+477.99	0.1912	--	--
Application Type: MIS & Communication; n=6					
13	PSU vs Effort	Y=5.5681x-303.86	0.7094	-4.58	↓
14	PSU_f vs Effort	Y=7.3699x-157.85	0.7499	31.67	↑
15	UFP vs Effort	Y=1.1943x+686.12	0.1351	--	--
16	Cfsu vs Effort	Y=0.594x+917.38	0.0393	--	--

and v1.21 results is an increase close to 17% (see in detail Annex E). The consequence on the results previously presented is in Table 2, updates previous results (UFP and Cfsu results are repeated for making easier the reading of results).

As evincible from the last columns, the new definition introduced in new PSU version returned improved results. In particular, it was noted an improvement using the solely PSU_f part both on MIS projects (+47.10%) as well as for MIS & Communication ones (+31.67%). But also looking at the overall dataset the improvement was notable (+7.21%). On the opposite side, two lower results were noted for MIS & Control projects (-34.01%) and MIS & Communication projects (-4.58%). In order to confirm such first-level results, further validations on new datasets must be done in the near future.

5 Conclusions and Prospects

One of the first and more important activities in any project is the estimation phase. In the Software Engineering domain from the end of '70s on the usage of estimations based on a functional size unit is more and more applied. But the increasing amount of non-functional effort in software projects can reduce the probability to successfully use a fsu as the solely independent variable in a regression analysis. The evidence of such problems and limitation of FSMM is when dealing with new technologies (i.e. DWH, R/T, Web applications), where there is a proliferation of interpretation for the original counting rules.

Looking at Scope Management practices from other application fields, the usage of a 'project-level' size unit can be a possible solution to complement and/or overcome the value brought out from FSMM.

Project Size Unit (PSU) is a proposal emerged in 2003 and freely available, created firstly for internal improvements in estimation practices, intimately based on

your own organization historical data, but also available for external usage with an agreement between customer and provider on the weighting system to be adopted.

The paper has presented main outlines for such technique and relationships with two of the most used FSM methods, namely IFPUG and COSMIC FSM. A case study with 33 sample projects was presented, sizing them against IFPUG v4.2, COSMIC-FFP v2.2 and PSU v1.01 methods. The comparison of regression analysis among the three techniques revealed that the proposed size unit (PSU) allows to obtain better effort estimates at the higher SLC phases more than FSM units as IFPUG and COSMIC. The update of PSU counting rules with the newer PSU version v1.21 shown that such changes (counting all tasks as peer types) was right both looking from a conceptual project management viewpoint and at the obtained numerical evidences. In any case, further attention will be paid in analyzing the reasons why for 'MIS & Control' or 'MIS & Communication' projects results are worst.

Next steps will be a further experiment with new projects, using an automated PM tool including PSU algorithm for verifying also the pros & cons in adopting PSU as a project size measure, observing also the effort needed for using it as well as the level of acceptance and feedbacks from estimators in project teams.

Acknowledgments

We would like to acknowledge the two students helping us in counting the 33 applications, Veronica Rubio Rodríguez and Enrique David Fernández Sanz.

References

[1] Project Management Institute, A Guide to the Project Management Body of Knowledge, 3rd edn. (2004) ANSI/PMI 99-001-2004, ISBN 1-930699-45-X

[2] Victoria Government, SouthernScope (2007) (23-05-2008), http://www.egov.vic.gov.au/index.php?env=-innews/detail:m1816-1-1-8-s-0:n-832-1-0

[3] FISMA, NorthernScope (2007) (23-05-2008), http://www.fisma.fi/in-english/scope-management/

[4] Buglione, L.: Project Size Unit (PSU) - Measurement Manual, v1.21e (November 2007) (23-05-2008), http://www.geocities.com/lbu_measure/psu/psu.htm

[5] IFPUG, Function Points Counting Practices Manual (release 4.2), International Function Point User Group (January 2004) (23-05-2008), http://www.ifpug.org

[6] Abran, A., Desharnais, J.M., Oligny, S., St-Pierre, S., Symons, C.: COSMIC-FFP Measurement Manual, Common Software Measurement International Consortium, Version 2.2 (January 2003) (23-05-2008), http://www.lrgl.uqam.ca/cosmic-ffp

[7] Buglione, L.: Project Size Unit (PSU) - Measurement Manual, v1.01, Technical Report (October 2005)

[8] ISO/IEC, International Standard 14143-1 - Information Technology - Software Measurement - Functional Size Measurement - Part 1: definition of concepts (February 1998)

[9] ISO/IEC, International Standard 14143-1 - Information Technology - Software Measurement - Functional Size Measurement - Part 1: definition of concepts (February 2007)

[10] Albrecht, A.J.: Measuring Application Development Productivity. In: Proceedings of the IBM Applications Development Symposium, GUIDE/SHARE, October 14-17, 1979, pp. 83–92 (1979) (23-05-2008), http://www.bfpug.com.br/Artigos/Albrecht/MeasuringApplicationDevelopmentProductivity.pdf

[11] Buglione, L.: Some thoughts on Productivity in ICT projects, WP-2008-01, White Paper, v1.1 (March 2008) (23-05-2008), http://www.geocities.com/lbu_measure/fpa/fsm-prod-110e.pdf

[12] Paulk, M.C., Weber, C.V., Garcia, S.M., Chrissis, M.B., Bush, M.: Key Practices of the Capability Maturity Model Version 1.1, Software Engineering Institute, CMU/SEI-93-TR-025 (February 1993) (23-05-2008), http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr25.93.pdf

[13] ISO, IS 9000:2005: Quality management systems – Fundamentals and vocabulary, International Organization for Standardization (September 2005)

[14] Buglione, L., Abran, A.: ICEBERG: a different look at Software Project Management, IWSM 2002 in Software Measurement and Estimation. In: Proceedings of the 12th International Workshop on Software Measurement (IWSM 2002), Magdeburg, Germany, October 7-9, 2002, pp. 153–167. Shaker Verlag (2002), http://www.lrgl.uqam.ca/publications/pdf/757.pdf ISBN 3-8322-0765-1

[15] CMMI Product Team, CMMI for Development, Version 1.2, CMMI-DEV v1.2, CMU/SEI-2006-TR-008, Technical Report, Software Engineering Institute (August 2006) (23-05-2008), http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html

[16] ISBSG, Glossary of Terms, version 5.9.1, International Software Benchmarking Standards Group (28/06/2006) (23-05-2008), http://www.isbsg.org/html/Glossary_of_Terms.doc

[17] Biagiotti, C.: Migliorare gli aspetti di stima e pianificazione di un progetto attraverso la customizzazione di un tool OpenSource di Project Management, University of Perugia, Tesi di Laurea, Perugia, Italy (July 2007)

[18] Buglione, L.: Improving Estimation by Effort Type Proportions. Software Measurement News 13(1), 55–64 (2008) (23-05-2008), http://ivs.cs.uni-magdeburg.de/sw-eng/us/giak/SMN-08-1.htm

[19] Buglione, L.: Project Size Unit (PSU) – Calculation feature in Project Management tools - Requirements, v1.0, PSU-AU-1.00e (December 2006) (23-05-2008), http://www.geocities.com/lbu_measure/psu/psu.htm

[20] Buglione, L.: Tutto quello che avreste voluto sapere sui Function Point (e non avete mai osato chiedere!). In: GUFPI-ISMA meeting, Rome, Italy (May 6, 2008) (23-05-2008), http://www.gufpi-isma.org

[21] Rodríguez Ruiz E., Estudio estadístico de la conversión de mediciones de puntos de función IFPUG a COSMIC-FFP, University of Alcalà de Henares (Spain), Escuela Técnica Superior de Ingeniería Informática, B.Sc. Thesis (16/01/2007)

[22] Fernández Sanz, E.D.: Estudio Y Evaluación De Psu (Unidad De Medida De Proyectos) Y Estudio Estadístico De La Conversión De Mediciones Psu A Puntos De Función Ifpug, University of Alcalà de Henares (Spain), Escuela Técnica Superior de Ingeniería Informática, B.Sc. Thesis (12/06/2007)

[23] Rubio Rodriguez, V.: Estudio y Application de las PSU (Project Size Unit) para la planificación de Proyectos Software, University of Alcalà de Henares (Spain), Escuela Técnica Superior de Ingeniería Informática, B.Sc. Thesis (12/06/2007)

[24] Instituto Nacional de Administración Publica, Metodología MÉTRICA versión 3, TIC0529-01 (23-05-2008), http://www.csi.map.es/csi/metrica3/

Annex A: List of Acronyms

Acronym	Term / Definition
Ac	Activity
B.Sc.	Bachelor diploma
BFC	Base Functional Components
CMM	Capability Maturity Model
CMMI-DEV	CMM Integration for Development
COSMIC	Common Software Measurement International Consortium
CPM	Counting Practice Manual
DWH	Data WareHouse
F/Q/T	Functional / Quality / Technical
F/Q/T/O	Functional / Quality / Technical / Organizational
FISMA	Finnish Software Metrics Association
FP	Function Point
FPA	Function Point Analysis
FSM	Functional Size Measurement
FSMM	FSM Method
fsu	Functional Size Unit
FUR	Functional User Requirement
ICT	Information & Communication Technology
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IFPUG	International Function Point Users Group
ISBSG	International Software Benchmarking Standards Group
ISO	International Organization for Standardization
KPA	Key Process Area
LOC	Line Of Code
ML	Maturity Level
NESMA	Netherlands Software Metrics Users Association
NF	Non-Functional
NFR	Non-Functional Requirement
OU	Organizational Unit
PA	Process Area
PHD	Project Historical Database
PM	Project Management
PMI	Project Management Institute
PP	Project Planning
PSU	Project Size Unit
R/T	Real/Time
SME	Small-Medium Enterprise
SPP	Software Project Planning
STAR	Software Taxonomy Revised
Sw-CMM	Software Capability Maturity Model
TDI	Technical Degree of Influence
TLC	Telecommunication
UAH	Universidad de Alcalá de Henares
UFP	Unadjusted Function Point
UR	User Requirements
VAF	Value Adjustment Factor
WBS	Work Breakdown Structure

Annex B: Projects' Data – IFPUG FPA v4.2 & COSMIC-FFP v4.2

Id	IFPUG FPA v4.2											COSMIC-FFP v2.2				
Id	Appl.Type	#UR	#ILF/EIF	#EI	#EO	#EQ	UFP (data)	UFP (EI)	UFP (EO)	UFP (EQ)	UFP	E	X	W	R	Cfsu
01	Application	81	14	45	13	14	101	141	54	44	340	74	67	45	48	234
02	MIS	51	6	19	22	41	42	71	88	123	324	56	60	16	36	168
03	MIS & Ctrl.	18	12	36	12	17	92	162	66	66	386	49	112	36	72	269
04	MIS, Ctrl & Comm.	62	12	38	12	12	98	164	55	43	360	62	86	38	24	210
05	MIS, Ctrl & Comm.	54	14	12	93	8	98	40	372	24	534	46	81	10	54	191
06	MIS	38	14	44	6	15	107	132	24	45	308	136	63	87	50	336
07	MIS	63	7	17	15	21	49	72	82	84	287	21	105	16	86	228
08	Comm. & MIS	11	5	16	5	0	35	54	20	0	109	21	23	15	6	65
09	Comm. & MIS	42	19	48	15	16	131	189	64	54	438	97	79	123	97	396
10	MIS & Ctrl.	36	12	38	16	17	82	114	64	51	311	39	95	38	28	200
11	MIS	41	13	31	14	17	93	110	58	52	313	75	71	33	55	234
12	MIS	47	9	23	18	14	63	78	92	42	275	51	60	25	22	158
13	MIS	61	10	26	16	20	70	111	72	65	318	68	68	23	110	269
14	MIS	63	6	30	14	19	42	151	60	58	311	82	91	30	107	310
15	MIS	41	14	41	22	0	96	159	91	0	346	73	83	50	43	249
16	MIS	24	19	75	13	0	133	228	52	0	413	50	57	52	56	215
17	MIS	25	14	56	14	13	98	202	56	39	395	84	70	98	55	307
18	MIS	57	15	30	17	5	103	96	68	15	282	72	61	41	39	213
19	MIS	71	13	34	17	10	92	118	81	33	324	70	65	27	62	224
20	MIS & Comm.	56	19	33	10	21	123	169	49	27	368	114	70	35	29	248
21	MIS & Ctrl.	88	11	25	12	29	80	90	52	90	312	72	110	42	99	323
22	MIS	60	9	6	15	6	63	25	69	140	297	22	62	110	42	236
23	MIS & Ctrl.	45	14	23	20	2	101	110	90	6	307	43	42	42	137	264
24	Application	57	7	20	26	6	43	120	161	18	342	10	25	0	6	41
25	MIS	51	8	14	8	70	56	76	35	175	342	62	22	26	15	125
26	MIS	100	12	31	6	20	84	128	29	64	305	47	61	40	26	174
27	MIS	73	11	30	13	13	77	139	53	43	312	134	60	49	50	293
28	MISy control	38	11	18	9	28	77	86	45	97	305	37	57	33	68	195
29	MISy control	65	15	35	6	16	102	159	37	48	346	143	78	34	122	377
30	MIS & Comm.	46	11	22	19	13	77	101	82	39	299	72	93	20	82	267
31	MIS & Comm.	37	25	19	12	13	161	81	48	39	329	51	50	24	19	144
32	MIS & Ctrl.	23	19	17	9	13	133	59	37	41	270	29	29	21	65	144
33	MIS & Comm.	11	12	43	0	14	90	204	0	43	337	145	103	43	33	324
	Max										534					396
	Average										318,0					234,0
	Median										328,6					231,2
	Min										109					41

Annex C: Projects' Data – PSU v1.01 & Effort (Total and by SLC Phase, According to METRICA 3)

		Size		Effort	By SLC phase/process										
Id	#UR	PSU v1.01	PSU v1.01 (F)	Effort (m/D)	Plan	Feasibil.	Analysis	Design	Construct	Implement.	Maint.	Prj Mgm	Qual. Mg	Security	Config. Mgmt
01	81	282	136	1236	96	25	118	121	524	38	61	64	115	35	39
02	51	154	50	797	87	25	61	55	271	53	118	49	29	24	25
03	18	369	176	1752	121	25	154	151	742	26	61	91	150	94	137
04	62	309	160	1504	96	25	130	133	681	26	61	76	110	69	97
05	54	142	62	743	79	25	58	57	287	40	61	49	38	24	25
06	38	285	128	1388	109	25	118	122	527	74	61	76	110	69	97
07	63	171	75	1055	111	25	70	76	431	49	61	61	70	44	57
08	11	137	55	886	96	25	56	52	362	26	61	58	62	39	49
09	42	224	103	1308	126	25	92	98	543	59	61	70	94	59	81
10	36	132	56	810	96	25	54	53	335	26	61	52	46	29	33
11	41	213	94	1200	127	25	88	96	473	26	61	70	94	59	81
12	47	171	63	1260	115	25	70	68	481	64	61	79	118	74	105
13	61	309	177	1461	147	25	128	150	634	59	61	64	79	49	65
14	63	304	159	1565	149	25	126	132	667	59	61	70	110	69	97
15	41	222	97	1132	128	25	92	88	432	26	61	67	86	54	73
16	24	132	52	876	96	25	54	51	355	26	61	58	62	39	49
17	25	132	58	723	96	25	54	51	291	33	61	46	30	19	17
18	57	259	109	1835	210	25	106	104	767	114	61	88	142	89	129
19	71	358	97	2589	215	45	148	149	1492	115	61	76	110	81	97
20	56	276	132	1506	125	25	114	111	665	77	61	73	102	64	89
21	88	286	125	1451	126	25	118	122	557	90	61	76	110	69	97
22	60	204	101	1086	117	25	84	81	470	64	61	55	54	34	41
23	45	180	69	1036	96	25	74	71	399	54	61	64	78	49	65
24	57	288	151	1020	96	25	120	117	415	26	61	52	46	29	33
25	51	214	92	1353	125	25	88	90	534	69	61	85	110	69	97
26	100	137	44	1161	117	25	56	53	432	26	76	79	118	74	105
27	73	316	147	1647	133	25	132	137	685	26	61	88	142	89	129
28	38	183	87	493	17	8	54	44	229	10	12	50	28	27	14
29	65	301	161	950	17	8	126	127	434	75	12	62	36	35	18
30	46	213	87	1229	117	25	88	102	462	70	61	70	94	59	81
31	37	194	97	931	96	25	80	93	390	26	61	52	46	29	33
32	23	156	51	898	96	25	64	66	304	40	61	67	86	54	35
33	11	157	84	502	50	8	66	57	174	26	12	46	25	24	14
Max	100	369	176,60164	2589	215	45	154	151	1492	115	118	91	150	94	137
Avg	51,0	213,00	97,07	1161,00	111,00	25,00	88,00	93,00	462,00	40,00	61,00	67,00	86,00	54,00	65,00
Median	49,6	224,55	101,00	1193,42	109,94	24,06	92,15	93,27	498,33	49,03	58,73	66,15	82,73	52,24	66,79
Min	11	132	44	493	17	8	54	44	174	10	12	46	25	19	14

Annex D: Projects' Data – PSU v1.01 & Effort (Total, Abs by Nature/Task Type, by # Task: Type, Complexity, Nature)

	Size		Effort (m/d)								Tasks	By Type			By Compl			By Nature	
Id	PSU v1.01	PSU v1.01 (F)	TotI	F	NF	%F	%NF	M	Q	T	#task	M	Q	T	H	M	L	F	NF
01	282	136	1236	596	640	48,22%	51,78%	193	378	665	337	56	44	237	3	7	327	231	106
02	154	50	797	258	539	32,37%	67,63%	171	268	358	165	24	14	127	5	7	153	120	45
03	369	176	1752	835	917	47,66%	52,34%	334	524	894	434	68	56	310	4	8	422	304	130
04	309	160	1504	777	727	51,66%	48,34%	263	409	832	354	53	41	260	4	5	345	255	99
05	142	62	743	322	421	43,34%	56,66%	160	202	381	158	26	14	118	4	5	149	112	46
06	285	128	1388	624	764	44,96%	55,04%	274	409	705	332	53	41	238	6	6	320	232	100
07	171	75	1055	461	594	43,70%	56,30%	219	304	532	206	38	26	142	4	8	194	136	70
08	137	55	886	353	533	39,84%	60,16%	197	271	418	172	35	23	114	4	5	163	108	64
09	224	103	1308	600	708	45,87%	54,13%	258	378	672	268	47	35	186	6	7	255	180	88
10	132	56	810	345	465	42,59%	57,41%	175	225	410	156	29	17	110	4	5	147	104	52
11	213	94	1200	531	669	44,25%	55,75%	262	363	575	260	47	35	178	5	6	249	172	88
12	171	63	1260	462	798	36,67%	63,33%	285	432	543	242	56	44	142	5	7	230	136	106
13	309	177	1461	835	626	57,15%	42,85%	241	317	903	328	41	29	258	6	8	314	252	76
14	304	159	1565	817	748	52,20%	47,80%	288	409	868	348	53	41	254	6	7	335	248	100
15	222	97	1132	497	635	43,90%	56,10%	251	340	541	262	44	32	186	5	6	251	180	82
16	132	52	876	343	533	39,16%	60,84%	197	271	408	168	35	23	110	4	5	159	104	64
17	132	58	723	319	404	44,12%	55,88%	153	179	391	144	23	11	110	4	5	135	104	40
18	259	109	1835	770	1065	41,96%	58,04%	348	631	856	332	65	53	214	9	5	318	208	124
19	358	97	2589	702	1887	27,11%	72,89%	334	1454	801	392	53	41	298	10	6	376	292	100
20	276	132	1506	721	785	47,88%	52,12%	281	436	789	318	50	38	230	7	5	306	224	94
21	286	125	1451	634	817	43,69%	56,31%	286	439	726	332	53	41	238	8	4	320	232	100
22	204	101	1086	540	546	49,72%	50,28%	207	268	611	222	32	20	170	6	6	210	164	58
23	180	69	1036	398	638	38,42%	61,58%	219	337	480	220	41	29	150	4	7	209	144	76
24	288	151	1020	535	485	52,45%	47,55%	175	245	600	288	29	17	242	4	5	279	236	52
25	214	92	1353	582	771	43,02%	56,98%	303	409	641	272	53	41	178	6	6	260	172	100
26	137	44	1161	375	786	32,30%	67,70%	295	439	427	214	56	44	114	5	5	204	108	106
27	316	147	1647	764	883	46,39%	53,61%	338	521	788	384	65	53	266	5	5	374	261	123
28	183	87	493	234	259	47,46%	52,54%	87	152	254	191	47	35	109	1	2	188	103	88
29	301	161	950	508	442	53,47%	46,53%	103	309	538	359	59	47	253	4	2	353	254	105
30	213	87	1229	502	727	40,85%	59,15%	262	393	574	260	47	35	178	5	7	248	172	88
31	194	97	931	466	465	50,05%	49,95%	175	225	531	208	29	17	162	4	5	199	156	52
32	156	51	898	294	604	32,74%	67,26%	192	340	366	206	44	32	130	4	6	196	124	82
33	157	84	502	268	234	53,39%	46,61%	105	117	280	209	44	32	133	0	4	205	128	81
Max	369	176,60164	2589	835	1887	57,2%	72,9%	348	1454	903	434	68	56	310	10	8	422	304	130
Avg	213,00	97,07	1161,00	508,00	638,00	44,1%	55,9%	241,00	340,00	574,00	260	47,00	35,00	178,00	5,00	6,00	249,00	172,00	88,00
Median	224,55	101,00	1193,42	523,27	670,15	44,2%	55,8%	231,24	375,58	586,61	265	45,30	33,36	186,21	4,88	5,67	254,33	180,48	84,39
Min	132	44	493	234	234	27,1%	42,8%	87	117	254	144	23	11	109	0	2	135	103	40

Annex E: Projects' Data – PSU v1.21 & Effort (Total, Abs by Nature/Task Type, by # Task: Type, Complexity, Nature)

	Size			Effort (m/d)								Tasks	By Type			Compl. (T-tasks)			Compl. (QM-tasks)		
Id	PSU v1.21	PSU v1.21 (F)	Diff. % PSU	Totl	F	NF	%F	%NF	M	Q	T	#task	M	Q	T	H	M	L	H	M	L
01	342	234	17,5%	1236	596	640	48,22%	51,78%	193	378	665	337	56	44	237	1	4	232	2	3	95
02	172	125	10,5%	797	258	539	32,37%	67,63%	171	268	358	165	24	14	127	1	6	120	4	1	33
03	440	308	16,1%	1752	835	917	47,66%	52,34%	334	524	894	434	68	56	310	1	4	305	3	4	117
04	359	259	13,9%	1504	777	727	51,66%	48,34%	263	409	832	354	53	41	260	1	3	256	3	2	89
05	163	116	12,9%	743	322	421	43,34%	56,66%	160	202	381	158	26	14	118	1	3	114	3	2	35
06	339	237	15,9%	1388	624	764	44,96%	55,04%	274	409	705	332	53	41	238	3	3	232	3	3	88
07	212	140	19,3%	1055	461	594	43,70%	56,30%	219	304	532	206	38	26	142	1	5	136	3	3	58
08	177	111	22,6%	886	353	533	39,84%	60,16%	197	271	418	172	35	23	114	1	3	110	3	2	53
09	276	185	18,8%	1308	600	708	45,87%	54,13%	258	378	672	268	47	35	186	2	5	179	4	2	76
10	161	107	18,0%	810	345	465	42,59%	57,41%	175	225	410	156	29	17	110	1	3	106	3	2	41
11	266	176	19,9%	1200	531	669	44,25%	55,75%	262	363	575	260	47	35	178	1	4	173	4	2	76
12	249	140	31,3%	1260	462	798	36,67%	63,33%	285	432	543	242	56	44	142	2	4	136	3	3	94
13	336	258	8,0%	1461	835	626	57,15%	42,85%	241	317	903	328	41	29	258	2	6	250	4	2	64
14	356	254	14,6%	1565	817	748	52,20%	47,80%	288	409	868	348	53	41	254	2	5	247	4	2	88
15	268	184	17,2%	1132	497	635	43,90%	56,10%	251	340	541	262	44	32	186	1	4	181	4	2	70
16	173	107	23,7%	876	343	533	39,16%	60,84%	197	271	408	168	35	23	110	1	3	106	3	2	53
17	149	108	11,4%	723	319	404	44,12%	55,88%	153	179	391	144	23	11	110	1	3	106	3	2	29
18	341	214	24,0%	1835	770	1065	41,96%	58,04%	348	631	856	332	65	53	214	5	3	206	4	2	112
19	402	299	10,9%	2589	702	1887	27,11%	72,89%	334	1454	801	392	53	41	298	5	3	290	5	3	86
20	326	230	15,3%	1506	721	785	47,88%	52,12%	281	436	789	318	50	38	230	3	3	224	4	2	82
21	340	238	15,9%	1451	634	817	43,69%	56,31%	286	439	726	332	53	41	238	4	2	232	4	2	88
22	229	169	10,9%	1086	540	546	49,72%	50,28%	207	268	611	222	32	20	170	2	4	164	4	2	46
23	228	148	20,4%	1036	398	638	38,42%	61,58%	219	337	480	220	41	29	150	1	5	144	3	2	65
24	293	240	1,7%	1020	535	485	52,45%	47,55%	175	245	600	288	29	17	242	1	3	238	3	2	41
25	279	176	23,3%	1353	582	771	43,02%	56,98%	303	409	641	272	53	41	178	2	4	172	4	2	88
26	220	111	37,7%	1161	375	786	32,30%	67,70%	295	439	427	214	56	44	114	1	3	110	4	2	94
27	390	265	19,0%	1647	764	883	46,39%	53,61%	338	521	788	384	65	53	266	1	3	262	4	2	112
28	193	104	5,2%	493	234	259	47,46%	52,54%	87	152	254	191	47	35	109	0	0	109	1	2	79
29	363	257	17,1%	950	508	442	53,47%	46,53%	103	309	538	359	59	47	253	2	1	250	2	1	103
30	267	177	20,2%	1229	502	727	40,85%	59,15%	262	393	574	260	47	35	178	1	5	172	4	2	76
31	213	160	8,9%	931	466	465	50,05%	49,95%	175	225	531	208	29	17	162	1	3	158	3	2	41
32	212	128	26,4%	898	294	604	32,74%	67,26%	192	340	366	206	44	32	130	1	4	125	3	2	71
33	211	129	25,6%	502	268	234	53,39%	46,61%	105	117	280	209	44	32	133	0	0	133	0	4	72
Max	440	308	37,7%	2589	835	1887	57,2%	72,9%	348	1454	903	434	68	56	310	5	6	305	5	4	117
Avg	267,00	176,43	17,2%	1161,00	508,00	638,00	44,1%	55,9%	241,00	340,00	574,00	260	47,00	35,00	178,00	1,00	3,00	172,00	3,00	2,00	76,00
Median	271,00	184,65	17,4%	1193,42	523,27	670,15	44,2%	55,8%	231,24	375,58	586,61	265	45,30	33,36	186,21	1,61	3,45	181,15	3,27	2,21	73,18
Min	149	104	1,7%	493	234	234	27,1%	42,8%	87	117	254	144	23	11	109	0	0	106	0	1	29

Proposals for Increasing Benchmarking Data Quantity and Quality of Projects Measured in COSMIC

Harold S. van Heeringen[1] and Luca Santillo[2]

[1] Sogeti Nederland B.V., Lange Dreef 17, 4131 NJ Vianen, The Netherlands
harold.van.heeringen@sogeti.nl
[2] Agile Metrics, Via A. Pollastrini 7, 00062 Bracciano (RM), Italy
luca.santillo@gmail.com

Abstract. With the release of the COSMIC (Common Software Measurement International Consortium) measurement method version 3.0 in September 2007, the COSMIC Functional Size Measurement (FSM) method [1] has reached a stable and mature status. Almost 10 years after its inception, COSMIC has proven itself to be a valuable functional sizing method for a broad range of different software types and domains, including business applications, telecommunication software, real-time systems, and hybrids of these, with any kind of logical architectural structure. Even though many organizations worldwide have already adopted the method (now known briefly as COSMIC Function Points) in their operations, still a significant lack of external benchmarking data is perceived in the industry. For instance, organizations that are measuring in COSMIC have less projects to benchmark themselves to, within the well-known ISBGS (International Software Benchmarking Standard Group) benchmarking database [2] (currently at version 10), with respect to older generation measurement methods and measures, as IFPUG or NESMA Function Points.

In this paper the COSMIC Benchmarking Committee, led by the authors, will be introduced to the public and its goals and intents will be outlined. Topics covered in the paper are, among others, suggestions to improve the current ISBSG data collection questionnaire(s) for better usage, possibly higher data collection accuracy, and/or for compliance to the recently-issued topics of levels of decomposition and levels of granularity, and the possibilities to convert old generation measures (as IFPUG and COSMIC) to COSMIC measures for practical project benchmarking and estimation purposes.

Keywords: COSMIC, Functional Size Measurement, Benchmarking, ISBSG, size conversion.

1 Introduction

After about ten years since the first publication of the COSMIC Functional Size Measurement method, a major upgrade of the method has been released in September 2007. COSMIC claims that with this upgrade, the method will reach a mature and stable status and new major changes are unlikely to follow.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 17–25, 2008.

In this paper, one of the major challenges of the COSMIC method is discussed. Next to the marketing issue (There are still a lot of people/organizations worldwide that could benefit from the method, but are unaware of its existence), there is still not enough data in the main project data repositories to be able to benchmark projects sized in COSMIC effectively with the industry standards. COSMIC recognizes this challenge and has formed a Benchmarking committee in order to address this issue. In this paper, the Benchmarking committee (the authors) will explain the issues and propose different actions to overcome these. First, the main changes of the COSMIC method from version 2.2 to 3.0 are discussed, as it is necessary to understand these in order to understand the proposed changes to the ISBSG data collection questionnaire.

2 Important Changes in COSMIC Version 3.0

Experienced COSMIC users may wonder which are the differences in the method between version 2.2 and 3.0. Truth is, there are not many major differences, but there are some changed and new concepts. In this paragraph, the important changes are highlighted.

First of all, the documentation structure of the method has been changed. Version 2.2 consisted mainly of the COSMIC measurement manual [3], containing the core rules and concepts of the method, and the Business Application Guideline [4] which provides practical counting rules to apply the method in day-to-day counting practices.

The structure of COSMIC version 3.0 is as follows [5]:

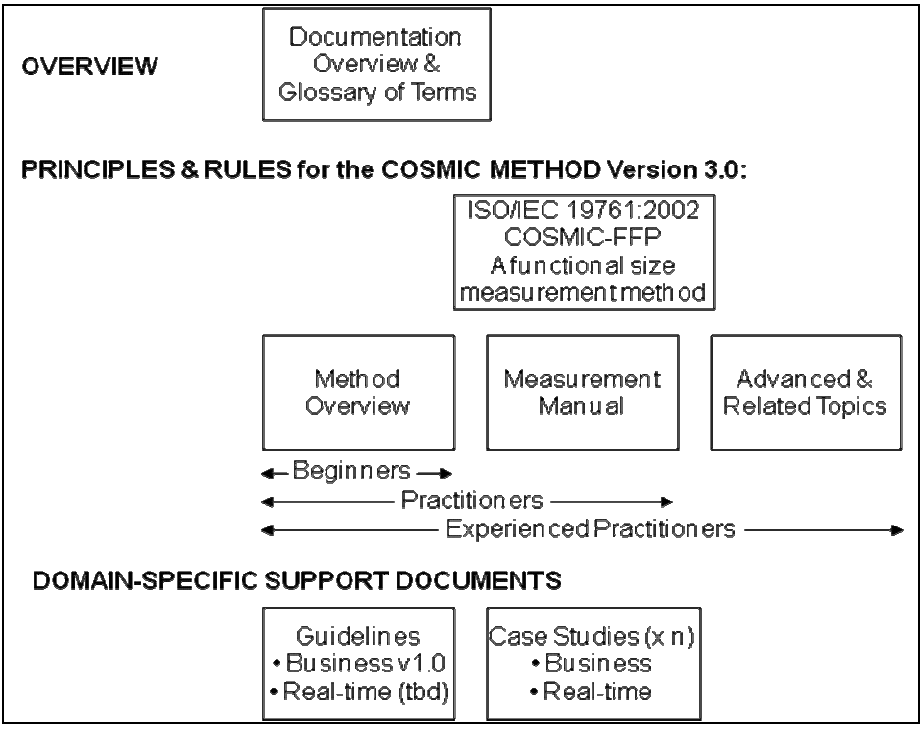


Fig. 1. Documentation structure of COSMIC version 3.0

In Version 3.0, the overview of the method's documentation and the glossary of terms is put into one document, the **Documentation Overview & Glossary of Terms** [5]. There are three documents that describe the Principles and Rules of the method. The **Method Overview** [6] gives an introduction to COSMIC and is relevant for beginners in functional sizing, as well as for practitioners in another FSM method. **The Measurement Manual** [7] is still the body of the method, containing all the detailed rules and principles of the method. In addition, there is a new document: **Advanced & related topics** [8]. In this document a number of relevant issues are addressed, like for instance early/approximate COSMIC sizing methods and convertibility possibilities between IFPUG and COSMIC size. **The Business Application Guideline** [4] still exists as a domain specific support document and also a number of helpful Case Studies have been published, which make it easier to understand the method as it is applied to real cases.

In addition to the documentation restructuring, also a number of changes have been made in the method itself. The size unit Cfsu (COSMIC Functional Size Unit) has been replaced by **CFP (COSMIC Function Point).** The name of the method itself has been simplified from COSMIC Full Function Points to just plain **COSMIC**.

Next to these cosmetic changes, also a number of more rigorous changes have been made, which lead to three new concepts: **functional users, level of granularity and level of decomposition.**

2.1 Functional Users

In version 2.2 of the COSMIC method, one out of two viewpoints had to be chosen: the developer viewpoint or the end user viewpoint. The main difference was that in the developer viewpoint the software could be segmented into multiple layers (like operating system, drivers, DBMS, etc), which made it possible to measure the software in the different layers seperately. In the end user viewpoint, the software to be measured was all considered to be present in one layer: the application layer. These two concepts have been removed from the method. They are replaced by the concept of the **functional user.**

The definition of a functional user is given as [7]:

'*A (type of) user that is a sender and/or an intended recipient of data in the Functional User Requirements of a piece of software*'.

A functional user can be therefore a human (end) user, but also other software or hardware systems. In COSMIC, the identification of the various functional users is derived from the purpose of the measurement. It is an important activity, as it helps defining the scope of the analysis.

2.2 Level of Granularity

The concept of Level of Granularity is introduced to make it visible that requirements can be specified on different levels of detail. The definition of the concept of 'level of granularity' is the following [7]:

'Any level of expansion of the description of a single piece of software (e.g. a statement of its requirements, or a description of the structure of the piece of software) such that at each increased level of expansion, the description of the functionality of the piece of software is at an increased and uniform level of detail.'

Early in the project lifecycle, requirements are usually specified with a low level of detail. During the project, the requirements will be refined and detailed to a level on which they show all the functional user requirements that the software must fulfill. The COSMIC Measurement manual [7] gives the following example:

'To illustrate the problems further, consider another analogy. A set of road maps reveals the details of a national road network at three levels of granularity.

1 Map A shows only motorways and main highways
2 Map B shows all motorways, main and secondary roads (as in an atlas for motorists),
3 Map C shows all roads with their names (as in a set of local district maps),

If we did not recognize the phenomenon of different levels of granularity, it would appear that these three maps revealed different sizes of the nation's road network. Of course, with road maps, everyone understands the different levels of detail shown and there are standard scales to interpret the size of the network revealed at any level. The abstract concept of 'level of granularity' lies behind the scales of these different maps.'

The COSMIC method recommends that one standard level of granularity should be used: the level at which all the functional processes and its associated data movements are visible and identifiable.

2.3 Level of Decomposition

The definition of 'Level of decomposition' in the COSMIC method is the following[7]:

'Any level of division of a piece of software showing its components, sub-components, etc.'.

This should not be confused with the Level of granularity concept described above. An example of different levels of decomposition, would be that an 'application portfolio' consists of multiple 'applications', each of which may consist of 'major (peer) components', each of which may consist of 're-usable object classes'. The various artifacts that have to be measured can be described on different levels of decomposition. Depending on the measurement purpose, a decision should be made at which level of decomposition the measurement should be made.

These important changes have to be taken into account when collection project data for benchmarking purposes.

3 COSMIC Benchmarking Committee, Goals and Initiatives

COSMIC has a traditional 'chicken-and-egg' problem. Most potential and new users of the method appear to want benchmark data, but few existing users seem willing to make the effort to submit data so that the existing benchmarks can be improved.

The COSMIC Benchmarking Committee is formed by both the authors of this paper and is appointed by the COSMIC organization in order to address the benchmarking issue described above. The goal of the Benchmarking Committee is :

To initiate initiatives to increase the number, and the data quality, of projects measured using the COSMIC Functional Size Measurement Method, available for benchmarking purposes.

The COSMIC Benchmarking committee has formulated a number of initiatives to support this goal. These are:

1. Update ISBSG data collection questionnaire to support COSMIC version 3.0 (addresses data quality)
2. Encourage COSMIC users data submission to ISBSG database (addresses data quantity)
3. Study size conversion methods in order to propose a method to convert the size unit of (part) of the ISBSG database from function points (FP) to COSMIC function points (CFP).

These initiatives are explained in the next paragraphs.

4 Update ISBSG Data Collection Questionnaire and Database Structure

The first initiative addresses the fact that the new version of the method also has its implications for the way the ISBSG collects data and the way the benchmark reports are produced. The ISBSG is a not-for-profit organization, based in Australia. Their claim is to be the global and independent source of data and analysis for the IT industry. Any project manager in the world can download a data collection questionnaire from the ISBSG website [1], fill in the data of a completed project, send it back to ISBSG and, in return, receive a free benchmark report for the project submitted. This way, the ISBSG database grows gradually and the people submitting data receive a reward in return. There are questionnaires for 'Software developments & Enhancement projects ', 'Software maintenance & support projects' and 'Software package acquisition & implementation projects'.

The current ISBSG data collection questionnaire (DCQ) for Software developments & Enhancement projects measured in COSMIC [9] counts no less than 138 questions. One of the issues that was already addressed in the ISBSG workshop (2007, Madrid) is the length of the questionnaires and initiatives have been undertaken since to lower the hurdle to submit projects to ISBSG. The COSMIC Benchmarking Committee has proposed shorter DCQ to ISBSG, in which the questions will be subdivided into three categories (essential/ important/ nice-to-have). This new questionnaire will probably be introduced in 2009. Another initiative to lower this hurdle is to develop a web-based data submission application. ISBSG is currently building this application. The online Maintenance & Support data entry tool is available since August 2008.

COSMIC version 3.0 requires a number of changes in the ISBSG data collection questionnaire in order to be able to benchmark projects in a sensible way against the data. The proposed changes are the following:

First of all, the name conventions have to be updated, so COSMIC-FFP has to be changed to COSMIC and COSMIC Functional Size Unit (Cfsu) to COSMIC Function Point (CFP).

Furthermore, the measurement strategy has to be explained explicitly per project. The description of the measurement strategy is now a formal part of any COSMIC measurement. In theory (and also in practice) it is sometimes impossible to benchmark projects to each other that are measured with different measurement strategies.

The measurement strategy [7] consists of four parameters: purpose, scope, functional users and level of granularity. The purpose of the measurement is the most important parameter and the other three parameters are derived from the purpose.

The ISBSG Benchmark committee proposes to standardize classes of the four parameters. When Benchmarking is only possible to projects that have the same (or at least a similar) measurement strategy, it is important to make it as easy as possible to identify these projects in the database. An example of standardization could be a question/answer combination like:

Q: What was the purpose of the measurement? (multiple answers possible)

A1. Software estimation before project start-up
A2. Performance measurement after project completion
A3. Size input to issue Request for Proposal
A4. Sizing an application portfolio
A5. Other …….

Another example could be:

Q: Which functional users were identified for the measurement? (multiple answers possible)

A1. Human end-users only
A2. Human end-users and interacting applications
A3. Peer components in the application layer
A4. Software in other layers
A5. Other …….

The exact standardization of the four parameters is still being discussed in the various bodies that are involved. Possibly some decisions are made at the 2008 ISBSG workshop in Nanjing, China. The ISBSG questionnaire has to be updated with questions and answers to identify the different parameters of the measurement strategy of the submitted project. Ideally, also the associated tools, like the ISBSG Comparative Estimation tool, would have to be updated in order to select only COSMIC projects measured with the same (or similar) strategy.

For users of COSMIC version 2.2, it should still be possible to state the viewpoint from which the measurement has been carried out (end-user or developer). These viewpoints have been removed from version 3.0 and substituted by the concept of 'functional users'.

These proposals will lead to an increased data quality of projects measured in COSMIC in the ISBSG repositories.

5 Encourage Data Submission

There are a number of initiatives to encourage COSMIC usage and data submission. One of the initiatives is an invitation letter that is composed by the COSMIC Benchmarking Committee in joint effort with the COSMIC MPC and the ISBSG. In this letter, contacts are invited to submit data of projects sized in COSMIC to the ISBSG. In return, an analysis will be given of the performance of the submitted projects against the relevant peer groups in the ISBSG repositories. The idea is to send this letter to the intensive network of the members of the MPC. Hopefully, this letter can be send out to all known COSMIC contacts in the fourth quarter of 2008.

In order to increase the awareness of the COSMIC method, and also to lower the hurdle of using it, the COSMIC documentation has been (or is being) translated in a number of languages. In 2008, COSMIC Measurement manuals in Dutch, French, Italian, Japanese and Turkish are expected to be published. In these translated measurement manuals, a request will be published to the users of the method to submit data to the ISBSG.

The paper that you are now reading (together with the presentation), submitted to the IWSM/Metrikon conference, should also lead to the same goal. Dear reader(s), if you have project data of projects measured in COSMIC, please help us out and submit them to ISBSG. We would be very obliged!

6 Functional Size Conversion, the Latest Insights

A third initiative to increase the project data available for benchmarking is to convert size from IFPUG/NESMA size to COSMIC size. A number of studies have already been conducted in this field, the last of which is the van Heeringen study [15]. In this study 26 projects were double sized with NESMA FPA and COSMIC after which a conversion formula was calculated. The main differences between this study and the previous publications are the larger number of projects (26) and the fact that the sample contained a heterogeneous set of projects, from various companies, operating in various branches. Although this study involved the use of NESMA function points, there is no reason to suspect another result for IFPUG function points, as there are only some minor differences left nowadays between these two methods. The various studies are summarizes in the next table.

Table 1. A number of published size conversion studies summarized

Author / year	Formula	Correlation	N
Fetcke (1999) [10]	Y(CFP) = 1.1 (IFPUG) – 7.6	$R^2 = 0.97$	4
Vogelezang & Lesterhuis (2003) [11] [12] [13]	Y(CFP) = 1.2 (NESMA) – 87 Y(CFP) = 0.75 (NESMA) – 2.6 (<200 FP) Y(CFP) = 1.2 (NESMA) – 108 (>200 FP)	$R^2 = 0.99$	11
Desharnais & Abran (2006) [14]	Y(CFP) = 1.0 (IFPUG) –3 Y(CFP) = 1.36 (IFPUG-TX) +0 (Transactions only)	$R^2 = 0.93$ $R^2 = 0.98$	14
Van Heeringen [15]	Y(CFP) = 1.22 (NESMA FP) - 64	$R^2 = 0.97$	26

It is possible to use one of the formula's above to convert a certain subset of the ISBSG repository from IFPUG/NESMA size to COSMIC size. However, this can only be done with extreme care. As the size in first generation functional size methods (NESMA/IFPUG) is measured purely from the view of the (end)user of the software, the result of the conversion will be the size of the software residing in COSMIC terms in the 'application layer'. When the measurement strategy for a specific COSMIC measurement for instance contains the purpose to benchmark a particular development team's performance against the ISBSG database, with the software residing in different layers, then benchmarking the project result against a converted repository subset doesn't make too much sense. The same is true for benchmarking real-time projects against converted project data, as first generation functional size methods could not have been used to size these kind of projects. For every benchmarking activity it is therefore mandatory to first analyze whether the project results are comparable, and if so, whether benchmarking against converted project data makes sense.

Because of the distinct differences between first generations of functional sizing methods (IFPUG/NESMA) and COSMIC, it would be advisable for ISBSG to market the COSMIC repository as a separate product. A number of projects from the current repository can be converted, in order to make sure that the repository is filled to some extent. A separate field should however be introduced to indicate whether it was an original measurement or a converted measurement. The Project Delivery rates should of course also be recalculated accordingly.

This way, COSMIC users don't have to pay for large amounts of data measured in IFPUG or NESMA that is not really usable for them (unless they do the conversion themselves). For ISBSG, COSMIC and the COSMIC users there is the advantage that the repository can be tailored specifically to the COSMIC method and therefore increase data quality.

7 Final Remarks

COSMIC is a non-profit organization and was developed by a group of volunteers, with no commercial involvement, for the general benefit of software engineering. The COSMIC method has been made available free-of-charge to every organization in the world and everyone could have benefited from the method's use. COSMIC now hopes to get something in return from the industry.

Carrying out the initiatives described above, the COSMIC Benchmarking committee hopes to be able to increase the number of COSMIC projects in the ISBSG repository from 110 (ISBSG R10) to at least 250 in the next release.

References

1. ISO/IEC19761:2003, Software Engineering – COSMIC – A Functional Size Measurement Method, International Organization for Standardization - ISO, Geneva (2003)
2. International Software Benchmarking Standards Group, Estimating, Benchmarking & Research Suite Release 10, http://www.isbsg.org

3. Abran, A., Desharnais, J.M., Oligny, S., St-Pierre, D., Symons, C. (eds.): COSMIC FFP Measurement Manual (The COSMIC implementation guide for ISO/IEC 19761:2003), version 2.2 (January 2003), http://www.cosmicon.com
4. Lesterhuis, A., Symons, C.: Guideline for sizing business applications using COSMIC-FFP, Version 1.0, December, http://www.cosmicon.com
5. COSMIC Measurement Practices Committee, Documentation Overview & Glossary of terms, Part of the COSMIC Functional Sizing Method version 3.0 (September 2007), http://www.cosmicon.com
6. COSMIC Measurement Practices Committee, Method Overview, Part of the COSMIC Functional Sizing Method version 3.0 (September 2007), http://www.cosmicon.com
7. COSMIC Measurement Practices Committee, COSMIC Measurement Manual (The COSMIC implementation guide for ISO/IEC 19761:2003), Part of the COSMIC Functional Sizing Method version 3.0 (September 2007), http://www.cosmicon.com
8. COSMIC Measurement Practices Committee, Advanced & related topics, Part of the COSMIC Functional Sizing Method version 3.0 (December 2007), http://www.cosmicon.com
9. COSMIC Data Collection Questionnaire 5.1 (September 2007), http://www.isbsg.org
10. Ho, V.T., Abran, A., Fetcke, T.: A Comparative Study Case of COSMIC- FFP, Full Function Point and IFPUG Methods, Département d'informatique, Université du Québec à Montréal, Canada (1999)
11. Vogelezang, F.W., Lesterhuis, A.: Applicability of COSMIC Full Function Points in an administrative environment: Experiences of an early adopter. In: 13th International Workshop on Software Measurement – IWSM 2003. Shaker Verlag, Montréal (2003)
12. Vogelezang, F.W.: Implementing COSMIC as a replacement for FPA. In: 14th International Workshop on Software Measurement – IWSM- Metrikon 2004, Konig-Winsterhausen, Germany (2004)
13. Abran, A., Desharnais, J.-M., Azziz, F.: Measurement Convertibility: From Function Points to COSMIC. In: 15th International Workshop on Software Measurement – IWSM 2005, Montréal, Canada, September 12-14, 2005, pp. 227–240. Shaker Verlag (2005)
14. Desharnais, J.-M., Abran, A., Cuandrado, J.: Convertibility of Function Points to COSMIC: Identification and analysis of functional outliers. In: International Conference on Software Process and Product Measurement (MENSURA), Madrid (2006)
15. Van Heeringen, H.S.: Changing from FPA to COSMIC - A transition framework. In: Proceedings of the 4th Software Measurement European Forum (SMEF 2007), Roma, Italy, May 9-11 (2007), http://metrieken.sogeti.nl, http://www.iir-italy.it/smef2007 or http://www.metricsmatter.net

Quality-Driven Orchestration of Services

Martin Kunz[1], Steffen Mencke[1], Niko Zenker[2], René Braungarten[1], and Reiner Dumke[1]

[1] University of Magdeburg, Software Engineering Group
P.O. Box 4120
39016 Magdeburg, Germany
{makunz,mencke,braungar,dumke}@ivs.cs.uni-magdeburg.de
[2] University of Magdeburg, Business Informatics
P.O. Box 4120
39016 Magdeburg, Germany
nzenker@iti.cs.uni-magdeburg.de

Abstract. The importance of providing integration architectures in every field of application is beyond controversy these days. Unfortunately existing solutions are focusing mainly on functionality. But for the success of Systems Integration in the long run, the quality of developed architectures is of substantial interest. Therefore a framework for quality-driven creation of architectures is proposed in this paper. Besides these quality-oriented characteristic the usage of semantic knowledge and structured process descriptions enable an automatic procedure. Especially the combination of both is a promising approach.

Keywords: software quality, quality driven design, service orchestration.

1 Introduction

Due to manifold advantages of high-flexible infrastructures compared to monolithic products a lot of initiatives propose approaches for the integration of single components (e.g. services). Semantic metadata provides the basis for the automation of this process. But those approaches lack from a throughout consideration of empirical data. Either only functional requirements or single quality attributes are taken into consideration.

In contrast to existing approaches the presented framework reveals a holistic orientation on quality aspects. It combines semantic web technologies for the fast and correct assembly of system elements and quality attribute evaluations for making the best assembly decisions possible. Therefore complex quality models are considered as well as empirical evaluations. Furthermore different types of quality evaluation like simulation and static and dynamic software measurement are used. Combining them delivers a holistic quality view on components and the flexibility enables a quality improvement of the targeted system by the exchange of single components if the evaluation of their quality attributes decreases.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 26–35, 2008.

The presented general QuaD^2-Framework (Quality Driven Design) can easily be adapted to a lot of different fields of application, e.g. service-oriented architectures or enterprise application integration.

2 Related Work

The introduction of dynamic (loosely coupled) distributed systems, where for each subtask a set of semantically equivalent candidates can exist, leads to the problem of the choice of the most appropriate one. The corresponding algorithms and procedures base on some kind of description of the components functional and non-functional properties. These descriptions are commonly referred to as metadata, are formally expressed in form of ontologies and stored in databases or repositories.

In the field of Web Services, the following ontologies can be used to describe non-functional properties: Ontology Web Language for Services (OWL-S) [1], Web Service Modeling Ontology (WSMO) [2], Web Service Quality of Service (WS-QoS) Ontology [3] or WSDL-S [4] (not a standalone ontology, but a semantic annotation technique for WSDL 2.0 [5]).

Web services which are enriched with such semantic metadata are called Semantic Web Services [6]. There are many approaches to using semantic information for the composition of web service-based business processes and workflows.

Berbner [7] proposes an infrastructure for the execution of service-oriented workflows, where a proxy service uses corresponding heuristics to select the most appropriate candidate for each workflow step on the basis of the workflow plan. These heuristics relate to QoS (Quality of Service) parameters such as throughput, response time and availability. Similar approaches are introduced by Menascé and Dubey [8], Tian [3] or Sirin, Parsia and Hendler [9]. But these QoS parameters describe only single aspects out of many as e.g. defined in ISO/IEC 9126 [10]. In contrast to the QuaD^2-Framework one can't identify these approaches as being holistic. Furthermore they lack from a consequent update, reuse and provision of empirical information – quality assurance during the whole lifecycle is not supported.

The METEOR-S project (Managing End-To-End OpeRations for Semantic web services) [11] uses four types of Web Service semantics (data, functional, non-functional and behavior semantics) to determine the optimal set of services for a given abstract WS-BPEL process and to generate an executable process.

Maximilien and Singh [12] describe a formal model and an implementation of an ontology-driven multi-agent environment for service selection.

As shown, existing approaches mainly focus on the functional aspects or single QoS aspects – a throughout quality-driven methodology is still missing. Furthermore, a general framework, not only focusing to special technologies or special domains as the chosen one above, is also missing.

3 QuaD^2 Framework

In general the sub-processes of this empirical-based assembly process are the initialization, the feasibility check (checking the functional coverage), the selection

process based on empirical data, and the operation of the established application. Quality assurance is achieved by certain sub-processes that allow optimizations at initialization time as well as during runtime. Furthermore measurement sub-processes are performed to update evaluation data.

The major goal of the described core process is an architecture consisting of single services. Such a service contains metadata-annotated functionality.

In order to achieve the sketched goals a special process is developed below. Its major use cases are introduced in figure 1.

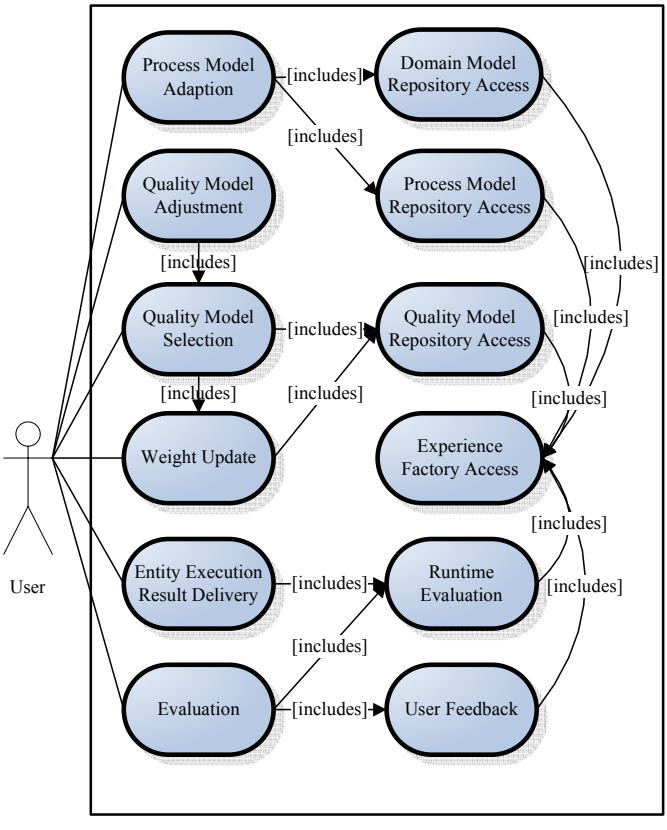


Fig. 1. Use Case Diagram: Empirical-Based Service Orchestration Process

The basis of the presented approach is a collection of semantically-annotated sources: the process model repository, the service repository, a quality model repository and furthermore an experience factory.

The process model repository is the source for process models that serve as descriptions for the functionality of the aspired distributed system. Example for such

processes can be ISO/IEC 15939 [13] for the software measurement process or didactical approaches [14]. Technological realization may vary, too. That can result for example in UML [15], BPMN [16], and ontologies [17].

An important source for empirical quality evaluations are quality models being provided by a quality model repository. The basis of a quality model's definition is an extensible list of quality attributes. The specification of a certain quality model is realized by selecting and weighting appropriate attributes. The evaluation and selection of appropriate services bases on evaluation criteria for each included attribute. Such attributes can be e.g. cost, performance, availability, security and usability. The attributes and corresponding evaluation formulas are standardized for example in ISO/IEC 9126 [10].

The service repository contains services, their semantic description and their evaluation data regarding all defined quality attributes.

The selection and adoption of process models and quality models are difficult tasks which constitutes the need for guidance and support. Because of this, the presented framework proposes the usage of existing experiences and knowledge about previously defined and used process models and quality models to support both process steps. Based on the Quality Improvement Paradigm, Basili and Rombach proposed the usage of an Experience Factory which contains among others an Experience Base and Lessons Learned [18], [19].

In the presented framework, the Experience Factory is fed from the process evaluation process and is the major building block to save empirical data and the user's experiences with specific process procedures or with distinct quality attributes.

Figure 2 defines the used diagram elements for the diagrams below. Optional elements have a grey border.

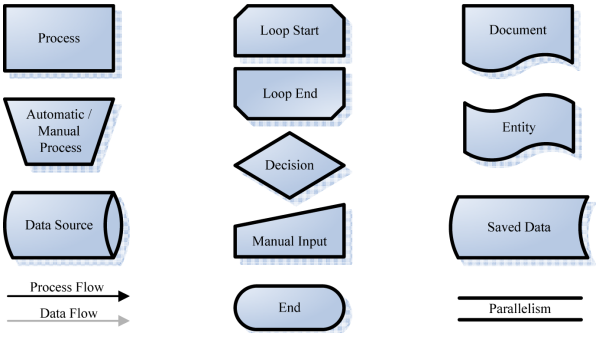


Fig. 2. Definition of used diagram elements

The focus on quality is a throughout property of the developed process and results in certain measurement and evaluation sub-processes that are introduced in the following general process description and are described more detailed in subsequent sections. The derived results are directly used for optimization purposes.

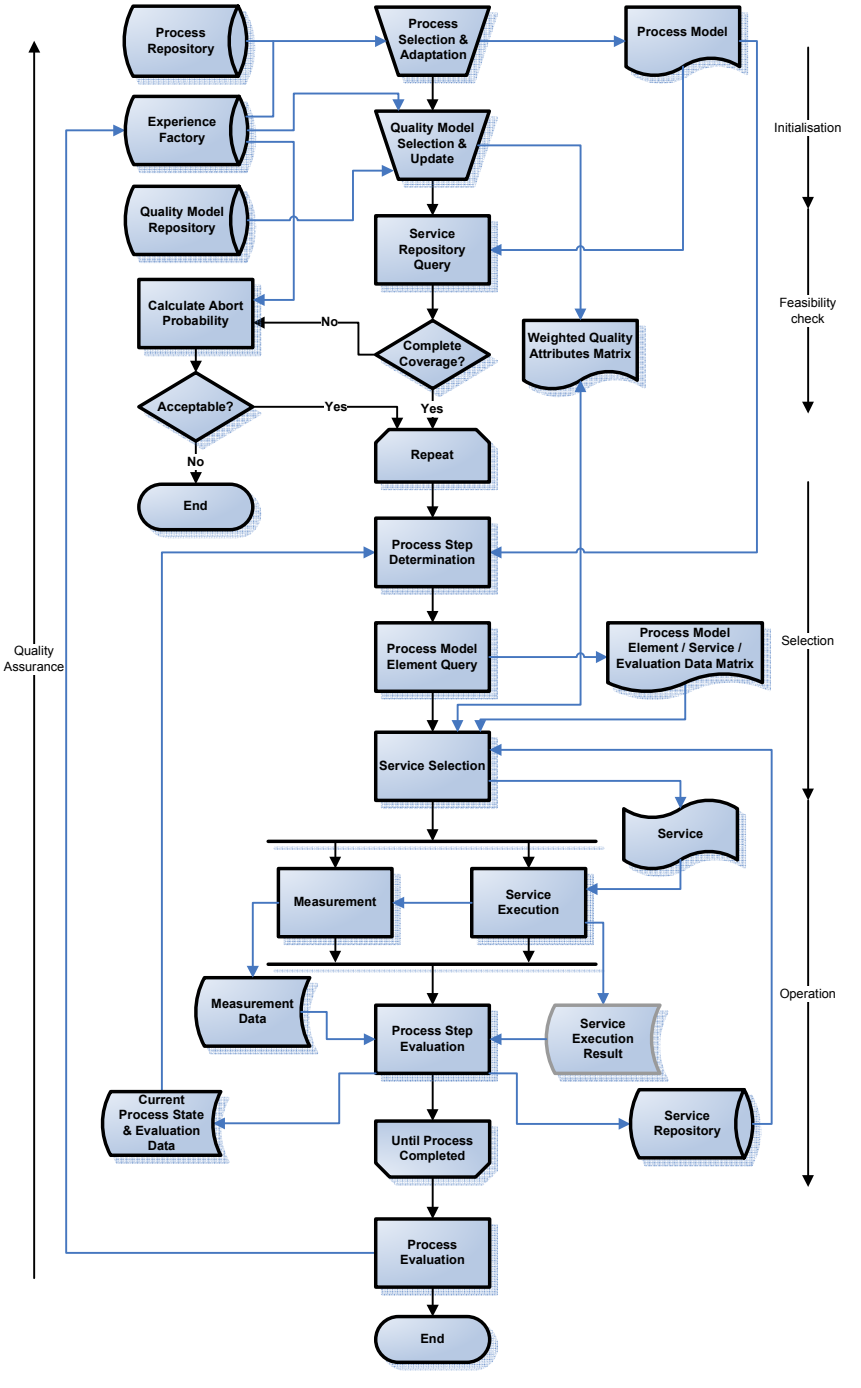


Fig. 3. QuaD^2 Framework

3.1 Initialization Steps

The selection of an appropriate process model that defines the functional requirements for the parts of the emerged distributed system is the first step. Due to the fact, that such a choice can be a manual process, it should be supported by an experience factory providing knowledge and experiences – lesson learned – for the decision for or against a specific process model for the current need. The process model essentially base on semantic metadata to allow the later automatic mapping of semantically described service functionalities to the functional requirements specified by the process model. With the chosen process model a set of concrete distributed systems is possible.

After the experience-supported selection of an appropriate process model the second step of the presented approach is a selection of a quality model from a quality model repository. This is intended to be done automatically. For certain domains manual adaptations can be more efficient. A manual individualization of this predefined set of quality attributes as well as of their importance weighting is also possible. For these purposes an experience factory can be helpful again. As a result of this step a process model and importance-ranked quality attributes are defined.

3.2 Feasibility Check Steps

With this information process step three is able to determine whether enough available services exist to provide an acceptable amount of functionality demanded by the process model. If there is no acceptable coverage after the negotiation sub-processes, then an abort probability based on already collected data can be computed. The user needs to decide whether he accepts the probability or not. If not the distributed system provision process will be aborted.

In the case of an acceptable coverage the runtime sub-processes of step 4 can start. The first of them determines the next process step to be executed following the process model. Therefore information about the last process steps can be taken into consideration to optimize the next process step execution. Exception handling in case of aborted pre-sub-processes is a functional requirement and thereby should be covered by the process model itself.

Due to the fact that new services can be added to the service repository, another coverage check for the next process step is performed next. Now, up-to-date service information, their evaluation values as well as the data of the quality model are available to identify the best service possible.

3.3 Selection Steps

The weighting of the quality attributes during the initialization delivered weighted attributes. This procedure is not intended to be performed during runtime, because the executed distributed system should not be interrupted (abort, costs …).

The result is a best possible distributed system based on the existing services as well as the specified quality model.

3.4 Operation and Evaluation Steps

Once the most optimal service is identified it can be executed and measured in parallel. These data are used to evaluate the last process step. The runtime sub-processes are repeated until either all process steps of the process model are successfully executed or an abort due to missing services took place. The last step five of the presented approach covers the evaluation of the entire process being an input for the experience factory. It compares the achieved results with the desired ones.

4 Quality-Based Service Selection Core Process

In general the service selection has several steps. The first identifies all possible services according to the required functionality defined within the process model (during initialization phase). An additional step selects the identified quality model that specifies what quality aspects are useful for the intended usage and how important they are for the initiator of the application to be assembled. Manual adjustments are possible, but not necessary and are performed during initialization, too. Only in exceptional cases a manual adjustment during runtime is reasonable.

Step three is the most important one and identifies the most appropriate service for the next process step to be performed during the selection process. It takes into account the weighted quality attributes as well the candidate service set whose elements fit the functional requirements of the current process step. Figure 4 shows a diagram presenting the underlying application flow of this special Service Selection Process.

The weighted quality requirements matrix is manually created by selection needed quality attributes from a predefined set during initialization. The user has to weight the attributes in a normalized scale. For example he can decide that the cost of a service has a weight of 70% (0.7), the performance is considered to be less important (20% = 0.2) and size is weighted with 10% (0.1). All weights must sum up to 1.

Amongst others the calculation formula and normalization directive are stored for all quality attributes to be able to determine the qualitatively best service for the current need.

Following the defined necessities and given data the service selection is formally described below. For the following formulas let PM be the chosen process model. Formula $f^{funct}(PM)$ specified in Formula 1 is used to determine the set of services E from the service repository. Each of them can deliver the functionalities specified within the chosen process model within formula 2.

$$f^{funct} : \text{Process model} \mapsto \{\text{Service}, \ldots\} \tag{1}$$

$$E = f^{funct}(PM) \tag{2}$$

Using the classic normalization approach presented in Formula 3, the evaluation values $v_{i,j}$ of quality requirements j defined in the quality model must be normalized

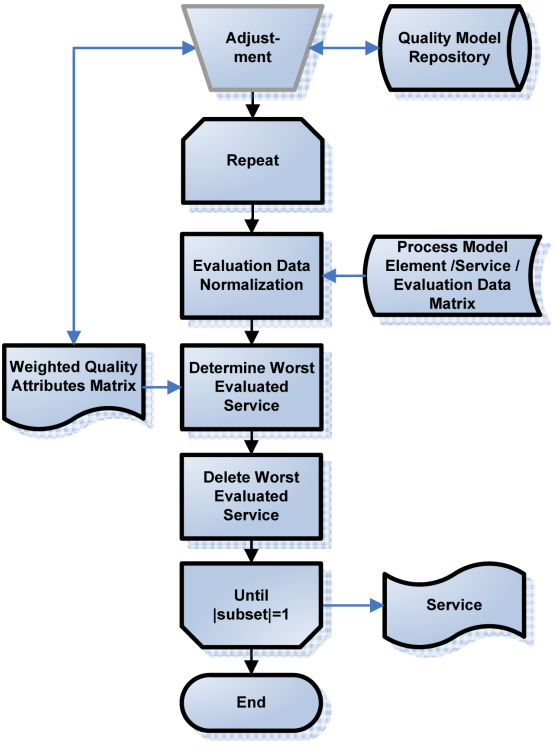


Fig. 4. Service Selection Process

for each service *i*. These $v_{i,j}$ are the measurement/simulation values to anticipate the optimal decision for the next process step.

$$v_i^{norm} = \frac{v_i - \min(v)}{\max(v) - \min(v)} * (\max_{norm} - \min_{norm}) + \min_{norm} \tag{3}$$

With the help of the weighted requirements matrix from the (maybe adjusted) quality model the last step – the identification of the optimal service according to the empirical data and the quality model – can be performed (see Formulas 4 to 8). Formula 4 adjusts the normalized evaluation values to ensure proper calculation. If *v=1* describes the best quality level then no adjustments are necessary, otherwise a minimum extremum is desired and *1-v* must be calculated.

$$f^{mm}(v) = \begin{cases} v & \text{, if a maximal } v \text{ is the best} \\ 1 - v & \text{, if a minimal } v \text{ is the best} \end{cases} \tag{4}$$

$$f^{eval}(e_i) = \sum_{j=0}^{n-1} f^{mm}\left(v_{i,j}^{norm}\right) \middle| e_i \in E \wedge n = |QM| \tag{5}$$

$$V = \left\{ f^{eval}(e_i) \middle| \forall e_i \in E \right\} \tag{6}$$

$$e^{worst} = e_{index} \mid index = \min(\{x \mid v_x = \min(V)\}) \wedge e_{index} \in E \tag{7}$$

$$E' = E / e^{worst} \tag{8}$$

To determine the best evaluated service, Formulas 5 to 8 are repeated until E' contains only 1 element. It provides the needed functionality and is the most appropriate one according to the specified quality model.

After the service's selection it can be executed and measurement about runtime behavior will be captured to get additional quality evaluations for this service.

5 Conclusion and Further Work

The presented framework provides a holistic quality driven procedure to aim high quality based design of distributed systems. The presented quality-driven approach uses semantic descriptions for processes automation and supports different quality models and quality attribute evaluations. The usage of standardized process models and standards for software quality and software measurement allows an objective evaluation of different service with comparable functionality but different quality attributes. The usage of the framework results in systems which are designed according to distinct quality requirements without affect the needed functionality.

The QuaD^2-Framework can be implemented using various technologies for example ontologies, web services and agents. The easy extensibility of process models, services, interfaces and quality models makes the presented framework deployable for many fields of application.

For the areas of software measurement infrastructures [20] first components are realized. Especially the ontologies and the empirical databases have to be filled. Therefore the main target is to integrate existing measurement databases or using existing metrics ontologies [21], [22]. The completion and usage of this task may reveal opportunities for future steps.

References

1. World Wide Web Consortium (W3C): Ontology Web Language for Services (OWL-S) (2004), http://www.daml.org/services/owl-s/1.1/
2. Fensel, D., Lausen, H., Polleres, A., de Bruijn, J., Stollberg, M., Roman, D., Domingue, J.: Enabling Semantic Web Services. The Web Service Modeling Ontology. Springer, Heidelberg (2007)
3. Tian, M.: QoS integration in Web services with the WS-QoS framework. Ph.D Thesis, Department of Mathematics und Computer Science, Freie Universität Berlin (2005)
4. World Wide Web Consortium (W3C): Web Service Semantics – WSDL-S. W3C Member Submission (2005), http://www.w3.org/Submission/WSDL-S/

5. World Wide Web Consortium (W3C): Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language W3C Recommendation (2007), http://www.w3.org/TR/wsdl20/
6. Studer, R., Grimm, S., Abecker, A. (eds.): Semantic Web Services. Concepts, Technologies, and Applications. Springer, Heidelberg (2007)
7. Berbner, R.: Quality of service support in service-oriented workflows, (in German), Ph.D Thesis, Technische Universität Darmstadt (2007)
8. Menascé, D.A., Dubey, V.: Utility-based QoS Brokering in Service Oriented Architectures. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2007) (2007)
9. Sirin, E., Parsia, B., Hendler, J.: Template-based composition of semantic web services. In: Proc. AAAI fall symposium on agents and the semantic web, Virginia, USA (2005)
10. ISO/IEC 9126-1:2001: Software engineering – Product quality – Part 1: Quality model (2001), http://www.iso.org
11. Aggarwal, R., Verma, K., Miller, J., Milnor, W.: Constraint driven web service composition in METEOR-S. In: Proceedings of the IEEE International Conference on Services Computing (SCC 2004), pp. 23–30 (2004)
12. Maximilien, E.M., Singh, M.P.: Multiagent System for Dynamic Web Services Selection. In: Proceedings of the AAMAS Workshop on Service-Oriented Computing and Agent-Based Engineering (SOCABE), Utrecht, Netherlands (2005)
13. ISO/IEC 15939: Information Technology — Software Engineering —Software Measurement Process (2002)
14. Mencke, S., Dumke, R.: Didactical Ontologies. Emerging Technologies in e-Learning 3(1), 65–73 (2008)
15. Object Management Group: Unified Modeling Language (OMG UML), Infrastructure, V2.1.2, Object Management Group (2007)
16. Object Management Group: Business Process Modelling Notation (BPMN), Object Management Group (2005)
17. Mencke, S., Dumke, R.: A Hierarchy of Ontologies for Didactics-Enhanced E-learning. In: Proceedings of the International Conference on Interactive Computer aided Learning (ICL 2007), Villach, Austria (2007)
18. Basili, V.R., Caldiera, G., Rombach, H.D.: The Experience Factory, pp. 469–476. Wiley & Sons, Chichester (1994)
19. Basili, V.R.: The Experience Factory: Packaging Software Experiences. In: Proceedings of the NASA Goddard Space Flight Center's 14th Annual Software Engineering Workshop, ISERN-99-19 Production and Maintenance of Software Measurement Models (1999)
20. Kunz, M., Schmietendorf, A., Dumke, R., Wille, C.: Towards a service-oriented measurement infrastructure. In: Proceedings of the 3rd Software Measurement European Forum (Smef 2006), Rome, Italy, pp. 197–208 (2006)
21. Martin, M., Olsina, L.: Towards an Ontology for Software Metrics and Indicators as the Foundation for a Cataloging Web System/ GIDIS, Department of Informatics, Engineering School at UNLPam, LaPampa, Argentina (2003)
22. Kunz, M., Kernchen, S., Dumke, R., Schmietendorf, A.: Ontology-based web-service for object-oriented metrics. In: Abran, A., Bundschuh, M., Büren, G., Dumke, R.R. (eds.) Proceedings of the International Workshop on Software Measurement and DASMA Software Metrik Kongress (IWSM/MetriKon 2006), pp. 99–106. Shaker Publ., Germany (2006)

Applying Six Sigma in the Field of Software Engineering

Ralf Russ, Dana Sperling, Frank Rometsch, and Peter Louis

Siemens AG, Corporate Technology,
Otto-Hahn-Ring 6, 81739 Munich, Germany
{Ralf.Russ,Dana.Sperling,Frank.Rometsch,Peter.Louis}@Siemens.com

Abstract. Process improvement in software engineering typically means introducing best practices. However, with increasing maturity of software engineering organizations the focus shifts from introducing industry best practices to optimizing the already implemented procedures and tools. But while in the field of software engineering a lot of best practice material exists, there is no proven and concise methodology for effective optimization.

The situation is similar to the one in the field of manufacturing some decades ago. They created Six Sigma, which is a problem solving and optimization methodology that is widely used today. But there are crucial differences between the disciplines manufacturing and software engineering. Whereas software artifacts are never developed twice and software processes are executed by humans, manufacturing is mostly machinery driven and usually deals with high quantities of identical output.

We applied Six Sigma in the field of software engineering and obtained promising results.

Keywords: Six Sigma, process, improvement, problem solving, optimization.

1 Introduction

At Siemens we have a 16 year history of using the SW-CMM®[1] and now CMMI®[1] to improve processes in systems, software and hardware engineering. During that period most of Siemens' operational units have implemented the CMMI® practices on Maturity Level 2 and 3, and also support these practices with integrated tools. By establishing effective engineering methods and processes, Siemens has gained significant business benefit from implementing CMMI®.

However, still not everything is as it should be. From time to time in special cases or environments implemented practices cause problems, although they were installed following the improvement guidance by CMMI®.

Even organizational units that are quite savvy with process maintenance and have quantitative process management and techniques such as root cause analysis in place,

[1] Software Capability Maturity Model (SW-CMM), Capability Maturity Model Integration (CMMI); SW-CMM and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 36–47, 2008.

often fail to effectively address those isolated problems. The reason is that the high maturity levels of CMMI® are typically implemented with a long term orientation. Their goal is to monitor overall process objectives as well as business relevant micro processes and to manage them. Moreover CMMI® does not provide any methodical or tool support; CMMI® is a process model and not a 'rapid reaction force'.

These kind of problems are exactly what Six Sigma addresses. Six Sigma is a methodology and toolbox for problem solving and improvement. It has been developed by Motorola in the 1980s with focus on application in manufacturing environments. This is also the area where a lot of Six Sigma experience exists in our company.

We have now introduced Six Sigma in the field of software engineering. So far we applied it to process problems in the area of peer reviews, code inspections, requirements engineering, latent defect management, sales and knowledge management, but also to product problems regarding performance and customer attractiveness.

2 Challenges of Applying Six Sigma in Software Engineering

Six Sigma encircles the problem thoroughly and builds up an As-Is model of the related system which is then analyzed qualitatively and quantitatively. The initial phase in a Six Sigma project always takes its time and makes people growing impatient. This is because of the best practice history we have in this field where we are used to directly jump on solutions.

A thorough investigation also encompasses quantitative analysis. Whilst manufacturing processes are highly automated, software processes intensively involve humans. Automated processes are fully defined and therefore observed variance is simply due to dispersion. Human based processes are never described exactly. This ambiguity contributes a considerable part to the observed variance. Moreover, in the software field, you will never develop the same module twice. Repetition is only achieved by mapping the work-results on some (few) attributes which also increases the dispersion part of variance.

Another difficulty in the field of software engineering is that one often has to deal with soft measures (e.g. estimation, rating, appraisal) instead of hard measures (e.g. time, length, weight). Data from soft measures is typically not continuous in its nature and thus less rich regarding the information one can draw from it. Therefore quantitative analysis is challenging in the field of software engineering and requires careful application of measurement and statistics.

One major lesson learned was that existing data often has not the required quality to be able to prove any conclusions. By applying the Six Sigma tools for measuring, the required data quality is determined upfront and the measurement procedure is designed to deliver data in the specified quality.

3 Examples

We present three examples which give a good overview on the experience we made. Two examples are concerning processes, one example is on improving a software product.

3.1 Knowledge Management

Within a software development consulting department of Siemens AG there was serious pain concerning knowledge management. The retrieval of acquired knowledge was felt to be bad, regarding effectiveness and efficiency, which led to double work, unprofessional appearance towards customers, or other undesired effects.

It was not the first time this department worked on its knowledge management process, but all attempts so far had no long term success. This time they joined the evaluation program and launched a Six Sigma project.

In the DEFINE phase the goals were acknowledged to be:

(1) Increase retrieval effectiveness (reduce retrieval defects) by 50%
(2) Increase retrieval efficiency (reduce time for information retrieval) by 30%
(3) Retrieval effectiveness must not depend on seniority.

Virtually no data exist in this area. It was decided to conduct a knowledge retrieval experiment.

The Y's (measure of effects) could be directly related to the goals. In the MEASUREMENT phase the X's (measure of controllable values) where derived by Ishikawa method. The measurement system was defined applying GQIM and CTQ-flow down. A measurement system analysis was done to verify the measurement operational definition.

The initial baseline was drawn by participating 17 persons from each section of the department and giving an appropriate representation regarding seniority. Each participant had to retrieve 10 knowledge items, which let to 170 data points. When a knowledge item had not been retrieved correctly, or could not have been retrieved within 4 times the expected mean retrieval time, it was declared as a defect of the process.

The quality of the data was really good. Retrieval time was lognormal distributed as expected from model considerations. Furthermore the retrieval process was stable, however with a lot of noise (s. Fig. 1).

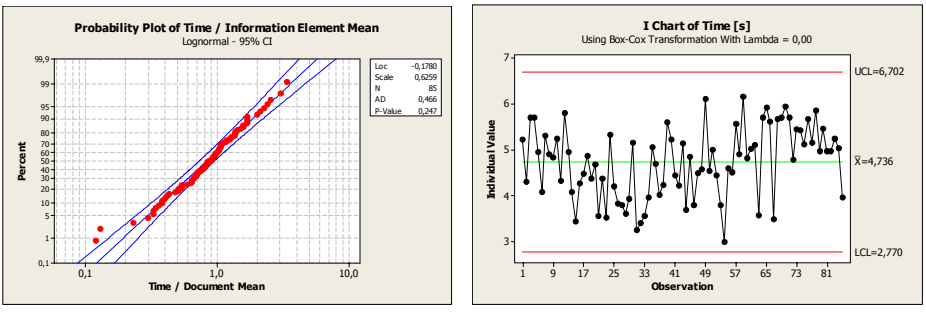


Fig. 1. The retrieval time shows a log-normal distribution. All data points have been included. However, the values of the data points have been normalized regarding the mean value of the knowledge item they belong to (left). The retrieval process is stable, but prediction quality is bad. The process shows a spread with factor 7. The logarithm of the retrieval time is plotted, as it is log-normal distributed (right).

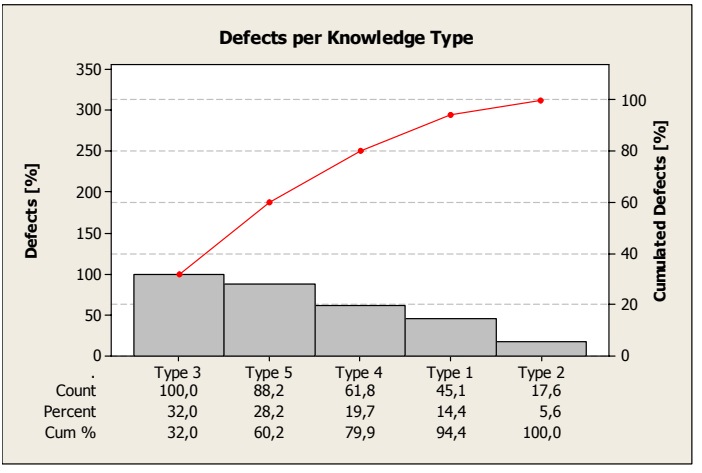


Fig. 2. The Pareto chart shows that more than 80% of all defects were due to three of the five knowledge types

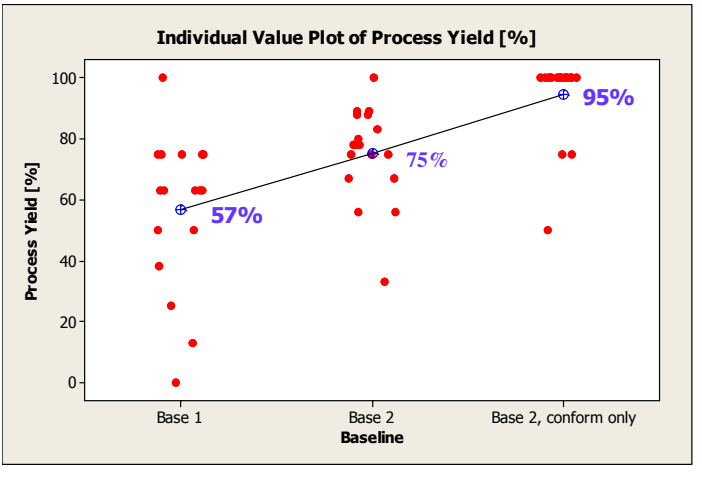


Fig. 3. The process yield of experiment runs increased significantly. Accounting only the process conform stored elements the process yield increased from 57% to 95%.

All knowledge items can be assigned to 5 different knowledge types. It turned out, that more than 80% of all defects occurred within three knowledge types (s. Fig. 2). Further analysis showed indeed that these knowledge types where differently stored and follow different retrieval processes. This picture was also supported when regarding the retrieval time. This was a really important learning from quantitative analysis and gave significant hint where to focus on and where to learn from.

As a result from ANALYSIS phase we could show that 14% (± 6%) of total work time was spent for information research and that furthermore retrieval effectiveness

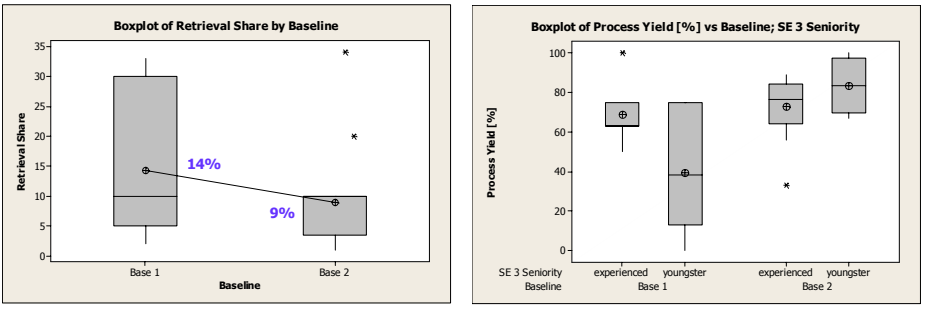


Fig. 4. Personal work time spent for information research has been reduced from 14% to 9% on average (left). The influence of the seniority of participants on their process yield has been eliminated by the improvement (right).

and efficiency depend significantly on the knowledge type and on the seniority of the people (s. Fig. 4).

Applying 5-Why analysis we identified 36 root causes for low effectiveness and efficiency of the knowledge management process. In the IMPROVEMENT phase these root causes and suitable 27 solution elements were assessed regarding damage and costs. The values came partly from the experiment and were partly estimated by the team. We brought this together in a QFD table which we called business model. It enabled us to determine the ROI for each solution element and gave us the prognosis for the total costs and fulfillment of our goals regarding the whole solution concept. This was an ideal basis for management decision.

The main changes concerned the structure of the knowledge base, the retrieval capabilities and the knowledge elicitation and qualification process. The accompanying change management and the management support were decisive for the successful implementation.

After training and roll-out of the new knowledge management procedure we spent half a year for practicing and adaptations. Then in the CONTROL phase we run a second knowledge retrieval experiment with 18 participants and 10 knowledge items. 8 out of 10 items where identical with the one from the first baseline. The participants were of cause different, but again representative for the department. It was charming, that 4 out of the 10 knowledge items were still not stored conform to the new knowledge management procedure.

Taking only the 8 identical knowledge elements into account, the average retrieval effectiveness (i.e. process yield) of the 17 respectively 18 experiment runs was increased by 42%. Accounting only the process conform stored elements the increase is 87% to an average process yield of 94% from 57% in initial baseline (s. Fig. 3).

The provable difference of the retrieval effectiveness concerning the seniority of the participants found in the initial baseline has been eliminated (s. Fig. 4).

The retrieval efficiency has been increased by 37% which means the amount of personal work time spent for information research has been reduced to 9%.

As this department is a Siemens-internal software engineering consulting group with about 50 persons, the business case was remarkable.

3.2 Peer Review Process

In a software business unit of Siemens people from both, management and project, complained about the performance of their review process. They worried that this obstructs the workflow of the projects and endangers the milestone dates. Reviews took their time and cost a lot of effort. Although several measurements on reviews existed nobody knew the real performance of the review process.

That was a real pain and they joined the Six Sigma evaluation program. In the DEFINE phase several six sigma tools were applied to investigate the problem in detail and to come up with a SMART project goal defined as:

Increase review process efficiency (reduce review effort by 15%) by keeping defect detection rate (review effectiveness) and fulfil safety relevant requirements.

Review process efficiency and effectiveness were the two big Y´s. Performing interviews with project members of 4 different projects using 5-Why method provided input for creating the initial process maps for the two major applied review processes - commentary and walkthrough reviews. These process maps described the processes as they are and were the basis to identify problem areas and corresponding X´s (see Fig. 5).

In the organization a measurement system existed containing historical data from the reviews of several projects. In the beginning we thought these are best conditions to go straight through MEASURE phase. But a first verification of the data discovered huge variation within data samples of 3 projects and unstable processes especially for commentary reviews. Assuming problems in the measurement system definition a Gage R&R study was done to evaluate current defect classification system. The result of this analysis made clear that the existing 5 defect classes were poly

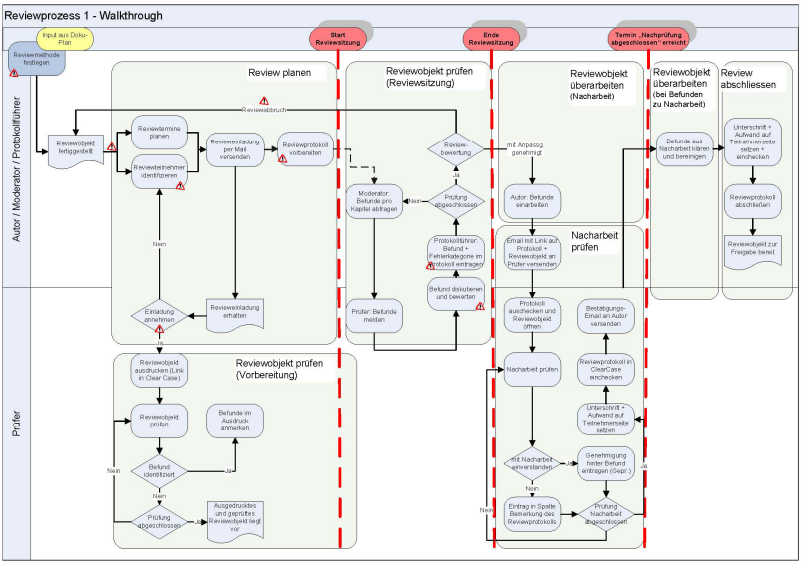


Fig. 5. Process map of walkthrough review

interpretable and far from being clearly distinguished. The data quality of X´s (especially number of defects per defect class and changed size of review document) was also very critical and the calculated baselines for review efficiency and effectiveness were questionable. Only the data sample of the walkthrough review process of one project had acceptable quality and the process was stable; for simplicity we assumed normal distribution, which could not be rejected (see Fig. 6).

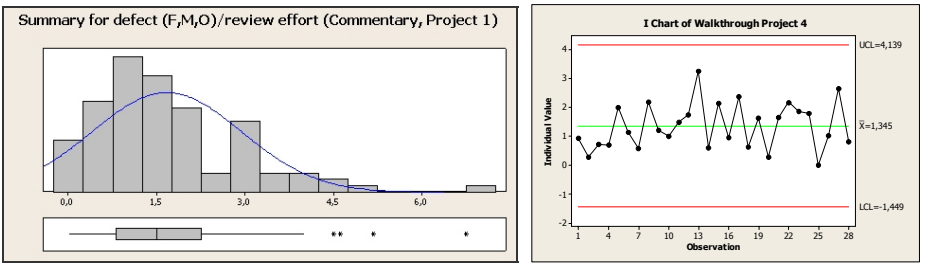


Fig. 6. The walkthrough review data shows a normal distribution. The process is stable.

At this stage of the project it became clear that we had to redefine the project goal. As the first step we had to improve the review process measurement system and to stabilize the review process.

In the ANALYSIS and IMPROVE phase of this redefined project we defined and prioritized appropriate improvement solutions based on the identified problem areas and influencing factors. For example we installed a new apparent defect classification system and we defined counting rules to ensure consistent filling-in of review records. We have also simplified this review record template to reduce effort for documentation. We redefined review planning process, adapt several methods and provide a checklist for review planning.

The altered review procedure was rolled-out and all users of the business unit have been trained.

Now we are in the CONTROL phase and the first new review data is available. This data shows a stable process and we could reduce variation by 45% (see Fig. 7). All new projects since July 2007 apply the changed review process and the new review measurement system. The new review metrics have high acceptance and are daily used to identify problems immediately.

3.3 Product Performance

As mentioned above Six Sigma is a problem solving and optimization methodology. Example 1 and 2 showed the application of Six Sigma to processes which enable developing complex software products. But can it also be applied to improve the product? During the execution of a software program, every little step follows dedicated rules defined by the code itself. The consecutive execution of these rules follows the same statistics as parameterized manufacturing lines. Therefore Six Sigma should also be applicable for software products. This project example of a business

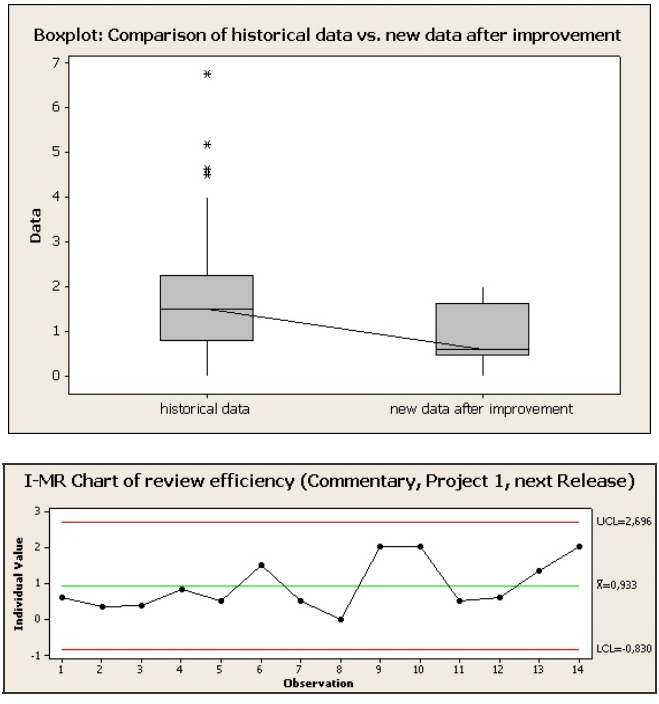


Fig. 7. The upper chart shows the improvement of review efficiency by 45% between the initial and the final baseline. The lower chart shows that the new review process is stable.

unit from the healthcare sector of Siemens describes such an application of the Six Sigma DMAIC methodology to a product to improve its performance.

The product itself is a so called picture archiving and communication system – called PACS. Whenever a physician puts a patient into imaging devices like x-ray machines or magnetic resonance tomographs, these devices produce a bunch of images from the inside of the patient's body. In the past all these medical images where stored as analog print out on so called films, you have to take with you when moving through the hospital. Nowadays these images can be electronically stored into PACS. These systems are not only used as archive, but do also coordinate and structure the workflow between several physicians normally needed to get a patient examined. Hereby the images can be accessed from everywhere and every time they need to.

A Six Sigma project begins with the perception of a problem. We heard from several different field trials, that the performance of our product, especially for one dedicated workflow, was not good enough. As several developers in the past did already work on that problem without achieving a long term solution, we decided to start a Six Sigma project to solve this problem.

The problem statement was more or less already defined by the customers. So we used the DEFINE phase to find CTQs influencing the criticized performance of the use case which is storing images into the archive and making them available for other physicians. And these two parts of this use case we could directly use as our Y's.

And what about the business case? As a result of the cost pressure in the healthcare sector, there is a trend to connect several hospitals to joint data clouds. To be able to share images across hospitals a high synchronization speed between connected sites is required. And therefore we wanted to use the same interfaces, customers already criticized in single-site installations. So the improvement was mandatory. Putting the ROI in numbers, an estimated project effort of 10 person-months stood against a business volume of over 80 million Euros of sales.

During define phase we also set our goal for this project, which we estimated to be an improvement factor of about 5 to reach the required performance for the mentioned use case and a dedicated, often used image type. A risk for reaching the goal was that this Six Sigma project needed to be time boxed as the achieved improvements should be part of the next planned product version, where the release date was already known by our customers.

After passing the first toll gate review, we started with the MEASURE phase. Using the two Y´s characterizing the two separated process steps, we discussed with domain experts what are the main influencing factors (the X´s). First we developed the overview as a Fishbone Diagram. In a second step we put the X´s into a list, added information about how likely it is, that they have significant influence on the process. Afterwards we grouped them according their feasibility to change their values or settings for different measurements. This list made it easy to derive out of it a Data Collection Plan, where we varied all easy-to-change parameters inside one measure-set automatically, whereas we did new test set-ups for the parameters where we e.g. need a reboot to change them.

So now we did know what to parameters to vary, but for the system under investigation we also defined some fixed parameters. For our system for example was the fill level of the contained database also seen as a important parameter which has surely impact on the performance. So before starting the measurements, we set up the system exactly in that configuration and amount of data in databases and on disks as defined upfront. For better repeatability and reproducibility we wrote some scripts for resetting the environment set-up. The measurement itself was done by an existing performance and load test suite, called PELOS. This test suite allowed us to do on the one hand the simultaneous testing with several clients against the archive, on the other hand it did all measurements from outside of the system. On the system we divided the process of storing images into the archive into 43 sub-steps, where for each of them developers added the measured process step time in milliseconds to existing Logfiles. Via a perl script these logfiles have been transformed to .csv lists so that we have been able to better use the data in statistic tools for further analysis.

Applying measurement system analysis we found out, that we need to increase the accuracy of the measurement system from milliseconds to microseconds. After the adaption of the measurement system to that finding, we took the first baseline from our system. But the measured values disappointed all team members, as they have been much worse then originally expected during define phase. Even if some test sets did meet the goals, some of them were so bad, that we need to improve the systems performance by a factor of 20! The reason for that was simply, that all measurements before the Six Sigma project did obviously not consider all important influencing factors good enough. But the structured definition of the measurement system, where you first think about what is necessary before you cut to what is feasible, helped a lot.

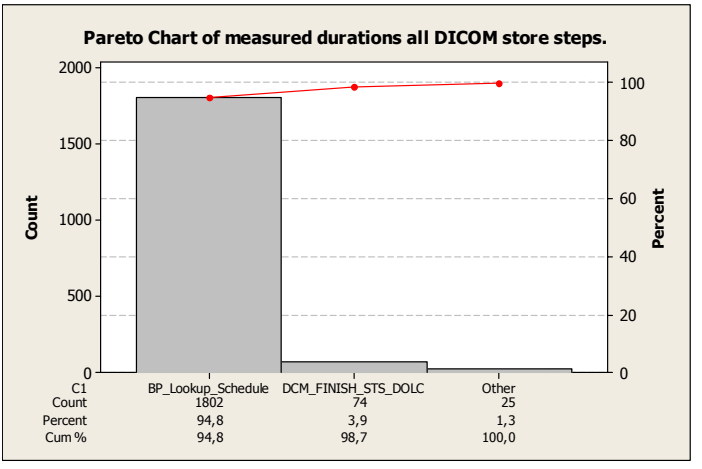


Fig. 8. Pareto chart of measured durations of all process steps

In the ANALYSIS phase we wanted to know, which of the process steps did consume all the process time. A Pareto analysis of all 43 process steps showed, that only one of it caused over 90% of the necessary process time (see Fig. 8).

Now it was clear, that before we analyze the system in all details, we first need to optimize this single step before, where our system fetches existing dataset objects and afterwards find and also Fuzzy-matches already existing attributes to the data set, which the system is currently about to store into the database. The experts discussed two possible solutions. First was to use the build in functionality of our database software, to self optimize the query performance by using statistics. The second solution was to optimize the data structure and queries manually, which brought better results and improved the system performance for some scenarios already by a factor of 10.

But as different datasets and load levels did still influence the performance significantly, we analyzed the system in detail. Interaction plots were particularly helpful to analyze afterwards the interaction of all relevant X's. In figure 9 one can see, that image size and number of parallel associations are still the most influencing factors. The smaller the image size, the lower the performance for the same data volume. And several concurrent associations also result in reduced performance. This effect is intensified when having small images.

In the IMPROVE phase another possibility to improve the performance even more was seen in varying one of the configuration parameter of the system. Therefore we did a Design of Experiment (DoE) to proof this. Compared to manufacturing processes or software engineering processes, where sampling is very often manual work, we had an almost fully automated measurement system in place. So measuring was not a very time consuming and expensive task and we opted for a DoE with full factorial design, where we measured the necessary transfer time when varying 3 parameters over 3 different values. Each test run had 10 replicates, so the sample size was in total 270.

As the results were non-normal distributed, we tried to transform these data of the measured transfer time to normal data, as for normal distributed data more analysis

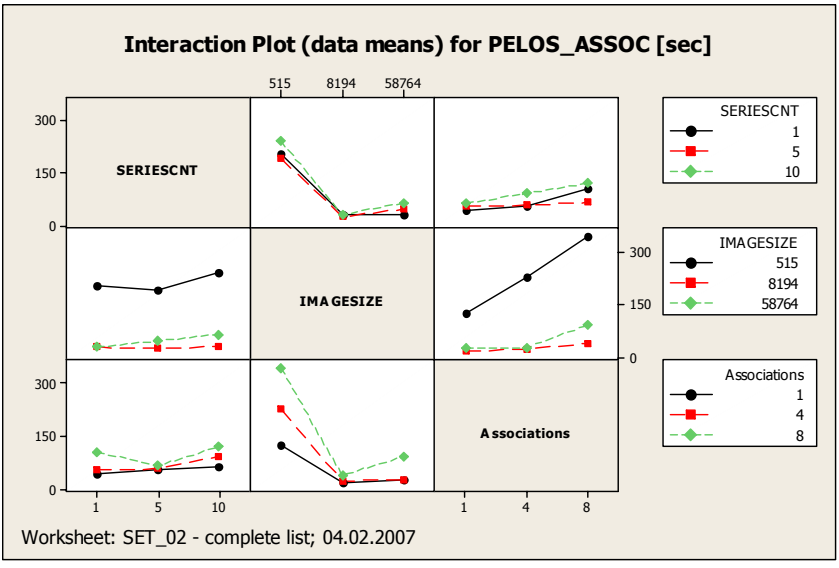


Fig. 9. Interaction plot of influencing factors

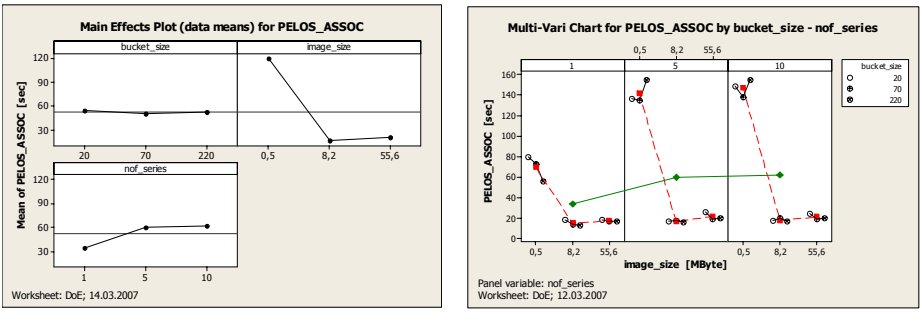


Fig. 10. Main-Effects plot and Multi-Vari-chart for analyzing the DoE

tools are available. But as the transformation via Box-Cox and Johnson transformation failed, this was a hint that we had in fact a multi-mode response behind, which got further analyzed by Multi-Vari-charts and Main-Effects-Plots.

In figure 10 on the left, the impact of our configuration parameter was not as big as expected by the experts. But as the two other parameters were still the most influencing factors, we decided to group the measured values based on these. In the used Multi-Vari chart (fig. 10 on the right) now the significant impact on some test scenarios got more clear, where the performance improvement was now up to factor 15,4.

The verification in the CONTROL phase showed in average an improvement of factor 15. The achieved goals were close to the target from define phase and the formerly worst performance inside our product portfolio turned into the best one. To sustain the achieved performance over the next software releases, regression tests

from the System test team have been extended to include the knowledge gained from this project.

As mentioned above, this Six Sigma project was time boxed and here we slowly ran out of time. But to give valuable advice for further improvement projects we used again Pareto charts to show, that 73% of the measured transfer time is caused by only 3 process steps out of 43.

In this project we spent most efforts in the definition and implementation of the measurement system (~30%) and the data collection itself (~30%). The reason is clearly not a presumable overhead through the Six Sigma approach itself, it rather shows how costly it is to produce good data – good enough to base decisions on it!

4 Summary

4.1 Benefits of Six Sigma Compared to Conventional Improvement

Applying the Six Sigma methodology is a means to systematic problem solving in the field of software engineering and leads to very high quality results.

Six Sigma enables decision-making based on facts instead of assumptions, and it gives strong confidence on the significance of conclusions. It forces to keep focus on the problem, it helps to identify the root causes, and to work out an effective solution. Six Sigma provides a powerful combination of both, qualitative and quantitative approaches, which complement each other.

Six Sigma is strongly focusing on the bottom line and enables improvement of processes and products with proven benefits.

4.2 Recommendations and Application Areas for Six Sigma

The results of our evaluation projects demonstrate that the Six Sigma methodology is applicable also in the field of software engineering. Its application area is the optimization of products or processes or solving of non trivial specific problems. One should keep in mind that working out effective individual solutions takes more effort than simply implementing industrial best practices addressing common problems.

We recommend to drive process improvements using the Six Sigma methodology if they are of high business impact or if no suitable best practices are available.

The CMMI® Maturity Level is neither restricting nor requiring the application of Six Sigma. However, organizations below Maturity Level 3 typically address common problems for which proven industrial best practices exist. Following the roadmap laid down by CMMI® is then the most efficient way of improving processes. While organizations on or above CMMI® Maturity Level 3 typically address specific problems without precedence, which then is the home ground of Six Sigma.

Acknowledgments. Thanks to Radouane Oudrhiri and Fabrizio Pellizzetti from Systonomy Ltd. who gave us valuable support.

First Steps towards Validating a Cost-Benefit Model of Reviews and Tests

Tilmann Hampp

Institut für Softwaretechnologie, Universität Stuttgart,
70569 Stuttgart, Germany
hampptn@informatik.uni-stuttgart.de

Abstract. Software project managers' decisions on reviews and tests are difficult. This paper describes a cost-benefit model for specific decisions on quality assurance. The quantitative model is based on single relationships and is quantified with historical data. Its results are shown and are compared with cost estimations. The model is able to reflect reported results of process improvement. Data collected in student projects is used to evaluate the model. Project averages and single projects are considered. Furthermore, results of a cross-validation are shown.

1 Introduction and Approach

Project managers have to decide on reviews and tests while planning and running a software project. But they do not have the necessary information. In particular, quality-related information is hard to get. Therefore, we developed a cost-benefit model which represents the decisions on quality assurance a project manager has to take. Costs and benefits are calculated for effort, time, and staff of individual activities. Long-term effects of quality affect corrective maintenance efforts and customer failure costs. For the purpose of comparing, the model results are summed up on a single scale in terms of money. The model reflects one project and its product and fits to a phase-based process with specification, design, and code. It is built on relationships [7, 12] and uses easily collectable metrics, in particular detected and corrected defects. The model enables one to run what-if analyses by varying quality assurance inputs. It complements cost estimations while planning and allows to analyse quality assurance afterwards.

This paper is structured as follows: Section 2 discusses related work. In Section 3, the model is described. Section 4 contains the model evaluation by data from student projects. Section 5 summarizes our conclusions and provides an outlook.

2 Related Work

Cost-benefit models for reviews consider saved effort for correcting defects [18, 28, 35]. Test models [8, 22, 32, 34] express benefit in terms of defects or reliability. The complete life cycle is considered in [37] based on the V-Modell XT. A generic simulation approach is used for modeling the IEEE-Standard 12207

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 48–61, 2008.

in [31]. Similarly, in [33], product-line development is simulated. In [13], costs and benefits of specific improvements in specific projects are assessed by a model. Our approach differs from these models regarding the considered benefit, and, in particular, the precise and detailed decisions on quality assurance. iDave [6] is built on COQUALMO and COCOMO II [5]. In order to calculate costs and benefits of quality assurance, the project's level of peer reviews, execution testing and tools, and automated analysis tools is rated. Our model uses COCOMO II to describe project and process attributes and uses the same relationships as COQUALMO. We focus on specific and detailed decisions and compute results bottom-up for single activities.

3 Model Description

The model computes costs and benefits of the specification review, the design review, the code review, and the system test. Figure 1 illustrates the model.

Each review and the system test are described by detailed inputs (Table 1). Inputs for overall quality assurance describe the combination of reviews, module test, integration test, system test, and field test, and whether they cover reused software. Tests during development and in maintenance can be repeated either not-at-all, or selectively, or completely. Additionally, process and product attributes are used, in particular the size of added, changed and reused software, as well as COCOMO II parameters.

The model considers costs for organising, preparing and executing reviews and the system test, as well as for rework and retest of detected defects. Defect detection is described by defect detection effectiveness (DDE) [26], distinguished

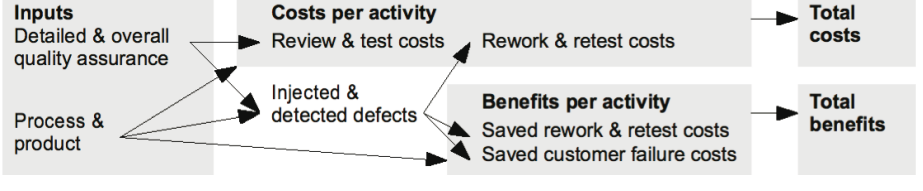


Fig. 1. Model overview and structure

Table 1. Detailed inputs for specification review and for system testing

Review	System test
Number of reviewers	Black-box test criteria: Coverage of functions, equivalence partitions, exceptional cases
Preparation rate	Glass-box test criteria: Coverage of statements, branches, conditions, loops
Reviewers' competence	Testers' competence
Document coverage	Retest strategy: not-at-all, selective, complete
	Preparation strategy: early stage, test phase

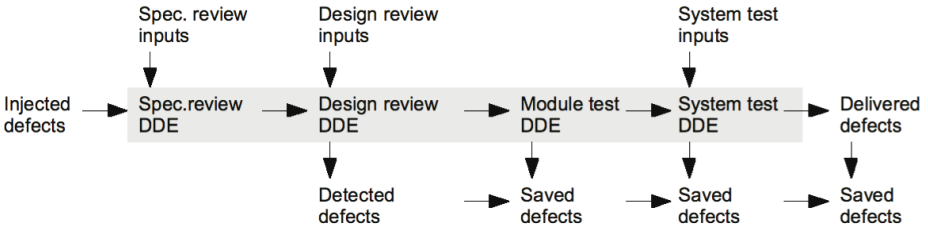


Fig. 2. Defect detection and saved defects for a design review

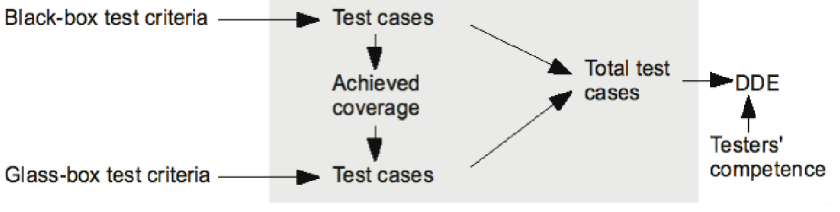


Fig. 3. Defect detection in system testing

by defect type (specification, design, code) and severity (minor, major, critical). Benefits are calculated as costs saved later on, i.e. in subsequent reviews, tests, and in maintenance, including customer failure cost. Figure 2 illustrates this for design reviews. Schedule and personnel are calculated from effort with COCOMO II. For comparing, effort, time and staff are weighted in monetary terms.

3.1 Model Details

Reviews. The inputs in Table 1 determine the model results: Defects can only be found in the covered part of a document. Each reviewer detects a fraction of the undetected defects. Defects are collected in a meeting, and duplicates are discarded [15, 17]. The preparation rate is used to describe that an intensive preparation is necessary [1, 17] but excess preparation will not reveal additional defects [1]. Additionally, defect detection is affected by reviewers' competence: A lack of knowledge leads to low defect detection [2]. Effort is calculated by the size of the covered document, preparation rate, and meeting rate, and is affected by low reviewers' competence. In addition to the reviewers, a moderator, the author and a recorder participate. Since large documents must be split [17], a gross period is calculated with three days between meetings.

System Testing. The number of test cases is essential for the model (Figure 3): Each test case detects an undetected defect with a certain probability [10, 14]. Each of the black-box criteria lead to a certain amount of test cases, and, after executing them, to covered code (Figure 3). To satisfy the glass-box test criteria, additional test cases are needed. Testers' competence is important because

defining test cases is creative work [29]. For each test case, effort is needed depending on the testers' competence. In black-box testing, effort for preparation, test harness construction and test case execution is distinguished because the test cases and the test harness can be prepared at an early stage [23]. Effort for preparing and executing a glass-box test can not be split up. But additional effort is needed for harness construction, in particular for instrumenting.

Costs and Benefits. Rework effort for a single defect depends on its severity [26], defect type and the defect's detection activity [4]. If the model input for retesting is set to "selective", a fraction of the test has to be repeated for every defect. Repeating a test requires a fraction of the effort for test execution [36]. Blocking defects are expensive [26] because they require retesting in any case.

Customer failure costs are calculated depending on the damage caused by one defect, its probability to cause this damage, and the usage frequency. These inputs are described by classifications. All defects are divided into these classes.

Organising efforts are considered by additional fractions. All efforts are affected by COCOMO II effort multipliers. Product size affects all efforts similarly to total project effort. COCOMO II is used to determine staff for each activity.

3.2 Realization and Quantification

Equations describe the model. E.g., with a given preparation rate r_p, the competence multiplier m_r and the number of reviewers n_r, a review's detection effectiveness is calculated with the parameters a_r and q_r by

$$DDE_\mathrm{review} = f(r_\mathrm{p}) \cdot m_\mathrm{r} \cdot (1 - a_\mathrm{r}(1 - q_\mathrm{r})^{n_\mathrm{r}}) \ . \tag{1}$$

Similarly, the system test's effectiveness is calculated for a given competence multiplier m_t and the number of test cases n_t by

$$DDE_\mathrm{test} = m_\mathrm{t} \cdot (1 - a_\mathrm{t}(1 - q_\mathrm{t})^{n_\mathrm{t}}) \ . \tag{2}$$

The achieved coverage c for different criteria is calculated for n_t test cases by $c = x \cdot (n_\mathrm{t}/n_\mathrm{n})^y$. The nominal number of test cases n_n standardises the test case number and is calculated from product size.

The model is quantified according to [5,24,25]. In particular, injected defects and their types are quantified as indicated by [24] for outsource projects. The equations above are quantified by in-depth studies for various defect types, tests and reviews (Table 2). The model is implemented as a spreadsheet.

3.3 Model Results

Comparing with COCOMO II. We compared the model results calculated bottom-up with results from COCOMO II. Model inputs were set for a nominal approach, including specification and design reviews with three reviewers each, module test, integration test and a typical black-box system test, covering functions and equivalence partitions.

Table 2. Parameter quantification for defect detection effectiveness

Parameter	Quantification
a_r, q_r	[1,24] e.g. 4 reviewers achieve a DDE of 55 % in specification review.
$f(r_p)$	Preparing intensively increases DDE up to 118 % [3].
m_r	Low-competence reviewers decrease DDE to 62 % [2].
n_t	0.6 test cases per function point are usual for black-box testing [25].
x, y	Black-box testing covers 50 % up to 60 % statements [19,34].
a_t, q_t	DDE varies from 35 % to 76 % depending on coverage [24,34,19].
m_t	DDE varies from 76 % up to 132 % [5].

Table 3. Comparing with COCOMO II results

COCOMO II effort (person-months)		Model costs (incl. correction)		Model benefit in project	after delivery
Plans & requ.	2.6	Spec. review	1.1	1.0	11.8
Product design	6.6	Design review	1.7	1.4	14.8
Programming	24.3				
Integration & test	8.6	System test	4.4	0.0	11.0

Table 4. Reported benefit by process improvement

Reported benefit	Model results (person-months) Total	Benefit	Remaining	Percentage
Rework dropped in integration to 20 % [20]		1.5	1.0	40 %
Retesting decreased to 40 % [20]	3.0	1.9		37 %
Rework decreased from 23 % to 14 % [11]	12.0	6.6		28 % to 15 %

The results for a small, 200-function-point project in Table 3 fit well. Effort for specification reviews seemed to be overrated by the model. Reviews nearly pay off during the project, but the main benefit is achieved after delivery. Schedule results are reasonable as well: Resulting in about 1 month for system testing and correcting, the model is in line with COCOMO II (3.1 months integration and test). Without reviewing the specification and the design, system testing stretches out to 1.5 months.

For comparing customer failure costs with review costs, we use a maximum damage per failure of 1000 €, e.g. one wasted working day. We assume that a defect causes a failure up to 10 times. A person-hour is rated with 100 €. The specification review costs roughly 16.000 €, but saves 76.000 € for the customer.

Comparing with Industry Experience. Haley et al. [20] report improvements due to CMM. In particular, reviewing design and code proved beneficial. We use the same model as above and consider design and code reviews. The benefit in integration testing is calculated too low (Table 4). Benefits in retesting

are calculated for the system test and fit well. Decreasing overall rework due to reviews [11] is estimated slightly too high by the model when all documents and code are reviewed. Since CMM contains other improvements and as model inputs are exemplary, we conclude that the results are reasonable. They are useful to demonstrate benefits.

4 Model Evaluation

We used data from student projects because we wanted to verify the model, e.g. identify missing relationships or unsound quantifications. We wanted to gain experience in calibrating and validating the model before using industry data. We aimed at assessing the model accuracy for describing actual data afterwards and for predicting data beforehand. A three-step approach is used:

1. Average values are used to verify the model.
2. Each project is compared with its individual model results.
3. Each project is compared with model results in a cross validation.

4.1 Student Projects

Second-year students majoring in computer science with a special focus on software engineering have to carry out projects in the Software-Praktikum [30]. In 2007, it went on from February to August. It was organised by tutors. One of them served as the project customer. He required a tool to define, document and manage test cases. 23 teams of 3 students each analysed these same requirements. Each team developed independently, following given milestones for specification, design, code, and associated quality assurance. The students had to review each others' specification and design. The teams had to do a module test and a system test. Each document was checked by the tutors. Finally, an acceptance test had to be passed.

Data Collection. First, required metrics were defined. Subjective ratings were subjected to counting rules. We collected the following metrics:

- Specification size and design size were measured in pages.
- Code size was measured in statements using CodeCount[1].
- COCOMO II cost drivers were subjectively rated for the projects altogether.
- The number and preparation effort of specification reviewers were documented by the review moderator.
- The number of test cases, the testing effort and the line coverage were documented in the system test protocol. Line coverage was measured using EMMA[2]. Black-box coverage for functions, equivalence partitions and exceptional cases were rated subjectively.

[1] http://sunset.usc.edu/research/CODECOUNT/

[2] http://emma.sourceforge.net/. We assume that line coverage approximates statement coverage.

- Effort for correcting defects, the detection activity, severity, and type (specification, design, code) of defects had to be documented by the students [21].
- Customer failure costs were rated subjectively by frequency, probability and cost per single defect.

Internal and External Validity. The lack of total-effort data and total-defect data is a threat to internal validity. Furthermore, the motivation and experience of the teams vary widely. We are not sure whether all defects were documented. Classifying defects is not always easy, although we prepared counting rules. These threats could lead to large errors and high dispersion. The same process with the same reviews and tests was required for all projects. Thus, model inputs for the projects differ only within a small range. This could make it difficult to validate the model's relationships because their effect could be hidden by uncontrolled variables.

External validity is threatened by the detailed and stable requirements, specification reviewers who analysed the same requirements, and students as participants, supported by tutors. Compared to industry projects, the product is small and simple. This could lead to systematic errors.

4.2 Model Results for Project Averages

The model parameters are set to the median code size (7104 statements), COCOMO II cost drivers and customer failure costs. Additionally, quality assurance inputs are set in accordance to the Software-Praktikum:

- The number of reviewers for the specification review and the design review is set to four. The preparation rate is set to 10 pages per hour.
- The module test is set to be carried out.
- The integration test is set to be carried out because the teams continuously integrate and test their code.
- System testing is set to complete functions and equivalence partitions. Statement coverage is set to 86 %.
- Reviewers' and testers' competence is set to nominal.
- The field test is set to be carried out, replacing the acceptance test.

First Results. Total effort, time and personnel are overestimated by COCOMO II (Table 5). The same goes for total defects. The distribution on defect types differ. But the test case number is reasonable. Further analysis showed that the model underrates defect detection in specification and design reviews. Results for design defects in testing were slightly too low. For all defect types, the detection in the acceptance test was overrated. The rate for correcting defects late did not rise by 10 : 1 but by 4 : 1 [21]. Specifications and designs were relatively small. The model's preparation rate for specification reviews is slower than the students' average rate (17.9 pages per hour).

Table 5. Project totals and uncalibrated model results

Metric	Model results	Actual average values
Total effort (person-hours)	2639	720
Total time (working days)	227	105
Total staff (persons)	1.9	3.0
Total defects	270	74 (median: 51)
Type distribution (spec./design/code)	26 %/28 %/45 %	51 %/19 %/30 %
System test cases	80	91 (median: 68)

Model Changes and Results. We aimed at restricting necessary changes to empirically confirmed experiences. But for calibrating such a model, experience is lacking. Therefore, we hypothesise that calibrating effort, schedule, defect count and defect distribution is necessary. A productivity factor is common [5, 27] and was added to adjust effort for single activities and total effort. Similarly, a schedule factor was added. We assume that the low defect level is due to product simplicity and size. Because adjustments could not be traced back to experience, factors for total defects and for defect type distribution were added and adjusted. The same was done for specification and design size.

An assumed cause for the high defect detection in specification reviews is that teams review each other. Because all reviewers analysed and specified the same requirements, they have become experts. The model did not include a positive effect of reviewers' competence. Hence, the calibration was reworked. Some reviewers are more capable than others [2], affecting their productivity and defect detection. This difference is used to readjust the model, and reviewers' competence is set to "very high". Alternatively, this input could be split up [9]. We stick to a single parameter since not enough data could be found.

Regarding the design defects, our experience is that designing is a hard, defect-prone task for the teams. We assume that during coding and continuous integration, many design defects were detected, but not documented. Nevertheless, the model was not changed. Regarding the defect detection, the formal integration test of the model seems to be in line with a continous integration. The acceptance

Table 6. Detected defects per type after model calibration

Defects	Model results			Actual values		
	Spec.	Design	Code	Spec.	Design	Code
Specification review	34.6			34.8		
Design review	1.8	9.5		0.1	11.6	
Module test	0.0	0.4	6.0	0.0	0.3	7.6
Integration test	1.1	1.2	7.2	-	-	-
System test	1.2	1.3	7.2	0.7	0.2	9.2
Acceptance test	1.6	1.0	2.2	0.0	0.0	0.4
Others	-	-	-	1.9	1.9	4.8

Table 7. Effort for correcting defects after reviews and tests

Correction effort (person-hours)	Model results	Actual median	Actual average
after specification review	7.5	6.9	9.6
after design review	3.4	2.5	4.5
after module test	2.0	1.6	4.9
after system test	11.9	9.9	9.2

test of the Software-Praktikum is not comparable to the field test in the model. In hindsight, one would exclude the field test, achieving better results.

In small projects, effort rises less heavily for correcting defects late [4]. A straight proportional function was added to the model, using an increase of 4 : 1 for projects with 100 function points and 10 : 1 with 10,000 function points.

Relationships for the system test remained unchanged as well as the basic relationships for defect detection, reviews, costs and benefits. The changes are leading to reasonable results (Tables 6 and 7).

4.3 Model Results for Single Projects

To explore the model accuracy, we concentrate on rework effort and defects, setting the following inputs for each project:

- the code size and factors for specification size and design size,
- the productivity factor and the schedule factor,
- the defect type distribution,
- the number of specification reviewers and the preparation rate,
- black-box system testing criteria and statement coverage.

We assess model accuracy by the relative error $MRE = \left|\frac{actual-estimated}{actual}\right|$ and its average $\overline{MRE}$ [16,27]. An acceptable level is 25 %, but this error margin is achieved even with COCOMO II in at most 80 % of the estimations [5]. In addition, the Pearson correlation r is used.

Results. Table 8 shows a fair model accuracy. Whereas defect results differ from actuals by roughly 70 %, error for effort exceeds 100 %. The error is largest for correction effort in system testing, illustrated in Fig. 4. Results for test cases are better ($\overline{MRE} = 0.45$, $r = 0.6$).

We assessed the benefit using the specification review intensity. The 7 teams with the highest preparation effort per page form one group. The second group contains the 7 teams with the lowest effort for preparing. We expected that intensive specification reviews detects more defects, leading to fewer defects and decreasing effort for correcting in system test. We only use projects for which both actual values and model results are available. Hence, effort for correcting defects after system test is based on 3 superficially reviewed teams and 4 intensively reviewed teams. Nevertheless, the results indicate a visible benefit in real projects as well as in model results (Table 9).

Table 8. Relative error and correlation for single project results

		Spec. defects	Design defects	Code defects	Total defects	Correction effort
Specification review	$\overline{MRE}$	0.61				1.13
	r	0.47				0.30
Design review	$\overline{MRE}$		0.52		0.77	1.88
	r		0.51		0.39	0.58
Module test	$\overline{MRE}$			0.74	0.83	1.27
	r			0.69	0.74	0.47
System test	$\overline{MRE}$			0.54	0.69	2.10
	r			0.69	0.67	0.46

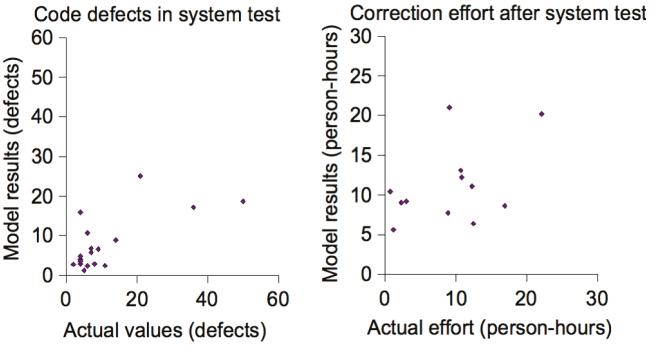


Fig. 4. Actual values and model results for system test

Table 9. Correction cost and benefit with specification reviews

	Spec. defects in spec. review		Spec. defects in system testing		Correction effort after system testing	
Averages for	Actual	Model	Actual	Model	Actual	Model
Intensive reviews	39.2	39.0	0.4	0.6	4.3	9.1
Superficial reviews	26.3	25.6	1.0	1.5	12.3	14.0

The model calculates that an intensive specification review leads to a benefit of 9.4 person-hours. Reviewing superficially, the benefit is 4.3 person-hours, resulting in 5.1 person-hours difference. Compared to the model's difference (4.9 person-hours) and the actual difference (8 person-hours) for correcting all types of defects, the results are reasonable.

4.4 Cross-Validation Results

We could not predict real projects beforehand. Instead, we use a cross-validation. All projects were split up in 10 groups. Results for each project were computed

Table 10. Relative error and correlation for cross-validation results

		Spec. defects	Design defects	Code defects	Total defects	Correction effort
Specification review	$\overline{MRE}$	0.60				1.39
	r	0.34				0.09
Design review	$\overline{MRE}$		0.76		1.01	2.98
	r		0.07		0.03	-0.05
Module test	$\overline{MRE}$			1.12	1.22	2.16
	r			0.54	0.58	0.71
System test	$\overline{MRE}$			0.65	0.79	3.05
	r			0.37	0.46	0.22

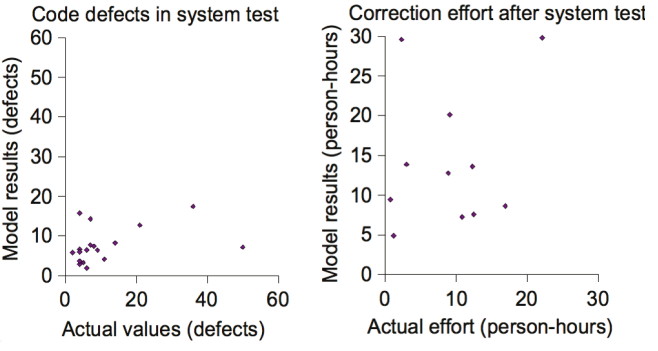


Fig. 5. Actual values and model results for system test

Table 11. Correction cost and benefit with specification reviews

	Defects in spec. review		Spec. defects in system testing		Correction effort after system testing	
Averages for	Actual	Model	Actual	Model	Actual	Model
Intensive reviews	39.2	34.7	0.4	0.6	4.3	10.5
Superficial reviews	26.3	27.3	1.0	1.9	12.3	12.0

using the other groups' average values for the productivity factor, the schedule factor, and the defect type distribution. Inputs for reviews and tests were set in accordance to project actuals. The results (Table 10, Fig. 5 for system testing) show increasing relative errors. The correlation is getting weaker. E.g., the average error for correcting after specification reviews is growing from 1.13 (Table 8) to 1.39 (Table 10), whereas the correlation is dwindling away (0.30 to 0.09).

In line with results for single projects, relative errors for effort are higher than those for defects. The relative error is largest for correction effort after system testing. However, the results for benefits of intensive reviews are roughly as

accurate as before (Table 11). On average, the model calculates a benefit of 8.9 person-hours for intensive reviews and 5.4 person-hours for superficial reviews, similar to the differences in correcting after system test.

5 Conclusions and Outlook

We conclude that the model reflects real projects in a qualitative, and, after calibrating, even in a quantitative way. Model flaws became apparent that led to model enhancements. We could hypothesise calibration parameters. We have calibrated and validated the model. Strictly speaking, validating the changed model using the same data is not possible. But one could argue that changes were restricted to independent, empirical experience.

The calibrated model accurately reflects average values. Model results for benefit reflect measured data. Results for single projects scatter widely, leading to a considerable relative error margin, but still reasonable results. Regarding the results of the cross-validation, we conclude that the more parameters are unknown, the less accurate the results become.

For us, this shows that projects are complex and affected by unknown variables. Hence, we have to expect an error margin on this fine-grained level of up to 100 % or even more. In light of this wide range, and as we are aware that complete data is seldom available, we address this uncertainty by varying parameter values. Then, the model is calculating result ranges.

The results indicate that the model is capable of demonstrating costs and benefits. It can be used to prove them afterwards. In project planning, the results should be complemented by other estimations, and we would suggest calibrating with historical data.

We are validating the model in two industrial settings. Additional tests are being incorporated in the model in more detail. We are going to complement the validation with a sensitivity analysis.

Acknowledgements. We would like to thank the participants of the Software-Praktikum and Sebastian Schumm for collecting the data.

References

1. Biffl, S.: Software Inspection Techniques to Support Project and Quality Management, Habilitationsschrift. Shaker Verlag (2001)
2. Biffl, S., Halling, M.: Investigating the Influence of Inspector Capability Factors with Four Inspection Techniques on Inspection Performance. In: Proc. of METRICS 2002 (2002)
3. Biffl, S., Halling, M.: Investigating the Defect Detection Effectiveness and Cost Benefit of Nominal Inspection Teams. IEEE Trans. on Softw. Eng. 29(5) (2003)
4. Boehm, B.W.: Software Engineering Economics. Prentice Hall, Englewood Cliffs (1981)

5. Boehm, B.W.: Software Cost Estimation with COCOMO II. Prentice Hall, Englewood Cliffs (2000)
6. Boehm, B.W., Huang, L., Jain, A., Madachy, R.: The ROI of Software Dependability: The iDave Model. IEEE Softw. 21(3) (2004)
7. Bossel, H.: Systeme, Dynamik, Simulation. Books on Demand (2004)
8. Cangussu, J.W., Mathur, A.P., Karcich, R.M., DeCarlo, R.A.: Software Release Control using Defect Based Quality Estimation. In: Proc. of ISSRE 2004 (2004)
9. Cuadrado-Gallego, J.J., Fernandez-Sanz, L., Sicilia, M.-A.: Enhancing Input Value Selection in Parametric Software Cost Estimation Models through Second Level Cost Drivers. Software Quality Journal 14(4) (2006)
10. Dahl, O.-J., Dijkstra, E.W., Hoare, C.A.R.: Structured Programming. Academic Press, London (1972)
11. Diaz, M., King, J.: How CMM Impacts Quality, Productivity, Rework, and the Bottom Line. CrossTalk (March 2002)
12. Drappa, A., Deininger, M., Ludewig, J., Melchisedech, R.: Modeling and Simulation of Software Projects. In: Proc. of the 20th Annual Softw. Eng. Workshop (1995)
13. El Emam, K.: The ROI from Software Quality. Auerbach Publications (2005)
14. Endres, A., Rombach, H.D.: A Handbook of Software and Systems Engineering. Empirical Observations, Laws and Theories. Pearson, London (2003)
15. Fagan, M.E.: Advances in Software Inspections. IEEE Trans. on Softw. Eng. SE-12(7) (1986)
16. Fenton, N.E., Pfleeger, S.L.: Software Metrics. A Rigorous & Practical Approach, 2nd edn. PWS Publishing Company (1997)
17. Freedman, D.P., Weinberg, G.M.: Handbook of Walkthroughs, Inspections, and Technical Reviews, 3rd edn. Little, Brown and Company (1982)
18. Freimut, B., Briand, L.C., Vollei, F.: Determining Inspection Cost-Effectiveness by Combining Project Data and Expert Opinion. IEEE Trans. on Softw. Eng. 31(12) (2005)
19. Grady, R.B.: Practical Software Metrics for Project Management and Process Improvement. Prentice Hall, Englewood Cliffs (1992)
20. Haley, T., Ireland, B., Wojtaszek, E., Nash, D., Dion, R.: Raytheon Electronic Systems Experience in Software Process Improvement. CMU/SEI-95-TR-017 (1995)
21. Hampp, T., Knauß, M.: Eine Untersuchung über Korrekturkosten von Software-Fehlern. Softwaretechnik-Trends 28(2) (2008)
22. Huang, C.-Y., Lyu, M.R.: Optimal Release Time for Software Systems Considering Cost, Testing-Effort, and Test Efficiency. IEEE Trans. on Reliability 54(4) (2005)
23. Jalote, P.: CMM in Practice: Processes for Executing Software Projects at Infosys. Addison-Wesley, Reading (2000)
24. Jones, C.: Applied Software Measurement. 2nd edn. McGraw-Hill, New York (1997)
25. Jones, C.: Estimating Software Costs. McGraw-Hill, New York (2007)
26. Kan, S.H.: Metrics and Models in Software Quality Engineering, 2nd edn. Addison-Wesley, Reading (2003)
27. Kemerer, C.F.: An Empirical Validation of Software Cost Estimation Models. Comm. of the ACM 30(5) (1987)
28. Kusumoto, S., Matsumoto, K., Kikuno, T., Torii, K.: A New Metrics for Cost Effectiveness of Software Reviews. IEICE Trans. on Inf. and Syst. E75-D(5) (1992)
29. Liggesmeyer, P.: Software-Qualität. Spektrum (2002)
30. Ludewig, J. (ed.): Praktische Lehrveranstaltungen im Studiengang Softwaretechnik. Bericht der Fakultät Informatik, Universität Stuttgart, 2nd edn. (2001)
31. Martin, R., Raffo, D.M.: Application of a Hybrid Process Simulation Model to a Software Development Project. Journal of Systems and Software 59(3) (2001)

32. Mizuno, O., Shigematsu, E., Takagi, Y., Kikuno, T.: On Estimating Testing Effort Needed to Assure Field Quality in Software Development. In: Proc. of ISSRE 2002 (2002)
33. Müller, M.: Analyzing Software Quality Assurance Strategies through Simulation. Dissertation, Fraunhofer IESE (2007)
34. Piwowarski, P., Ohba, M., Caruso, J.: Coverage Measurement Experience During Function Test. In: Proc. of ICSE 1993 (1993)
35. Rubey, R.J., Browning, L.A., Roberts, A.R.: Cost Effectiveness of Software Quality Assurance. In: Proc. of NAECON (1989)
36. Van Megen, R., Meyerhoff, D.B.: Costs and Benefits of Early Defect Detection: Experiences from Developing Client Server and Host Applications. Software Quality Journal 4(4) (1995)
37. Wagner, S.: Cost-Optimisation of Analytical Software Quality Assurance. Dissertation, TU München (2007)

Field Study: Influence of Different Specification Formats on the Use Case Point Method

Stephan Frohnhoff and Thomas Engeroff

Capgemini sd&m AG, Berliner Str. 76, 63065 Offenbach, Germany
{stephan.frohnhoff,thomas.engeroff}@capgemini-sdm.com

Abstract. The Use Case Point method (UCP method) allows early, easy estimation of the anticipated effort during a software development project. The basis for such estimation in real industrial projects is commonly a number of rough specifications in different formats and of differing granularity. The success of the UCP method and comparability of the results depend above all on whether and how good use cases can be identified and weighted from the specifications. Within a field study, a total number of more than 200 UCP estimations based on eight different specification formats have been performed. The estimations have been compared quantitatively and qualitatively with regard to the reproducibility of effort estimation and with regard to expert valuations. With the help of statistical methods a mean variance (variation coefficient) between 13 % and 48 % was found depending on the specification format. Thus, a valuation of specification formats for improving estimation accuracy could be derived with the help of variance analysis.

Keywords: project effort estimation, top-down estimation, use case points, UCP, specification, estimation reproducibility, field study.

1 Introduction

The Use Case Point method (UCP method) has already been described in detail in [1]. It is a top-down estimation method that classifies the functional requirements of the specification into countable units (use cases), to which points are assigned (use case points) according to their estimated complexity. The estimated project effort is then proportional to these use case points.

An important point of criticism about this method is that use cases can be described in different granularity, which could directly influence the UCP estimation result [2, 3, 4]. To reduce the influence of different formats a user guide to handle the different formats has been developed in the past [5].

To obtain an empirical evaluation of the reproducibility using the UCP method, a field study has been set up and executed [6]. The goal was to find out how big the intrinsic error of the UCP method is and how much the UCP method depends on the format of the specification the estimation is based on. Students at eight universities in Germany applied the UCP method on specifications provided in different formats. Thus, it was possible to analyze more than 200 UCP estimations with statistical

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 62–75, 2008.

methods. In addition, the results have been compared with estimations performed by experienced software engineers.

The context for this article and the field study are software development projects for business information systems delivered by the software house Capgemini sd&m as service provider for clients from different industries. The effort estimations are based on different rough specifications provided by clients or developed with clients and they follow no uniform format. The formats of the specifications vary from UML-like descriptions to purely text specifications.

2 Use Case Point Method

The UCP method has been described in detail in [1]. The original UCP method [7] reveals an insufficient standard deviation for industrial usage and an enhanced version has been developed [8]. The following analysis is based on this enhanced version called UCP 2.0 [6] which is described in the following. Figure 1 provides a schematic summary. The total effort of a software development project is defined by the effort caused by the functional requirements (A-Factor) times the technological factor (T-Factor), the management factor (M-Factor) and the productivity factor (PF).

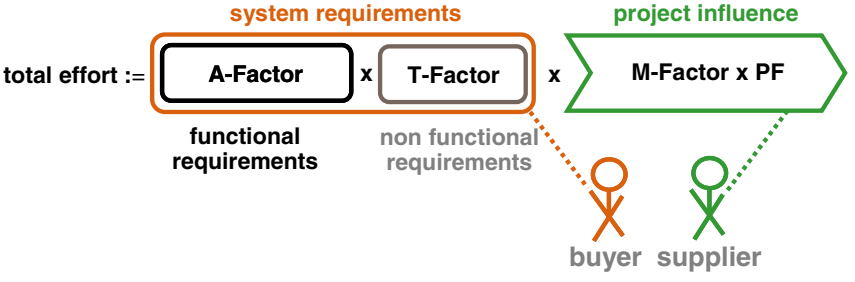


Fig. 1. Project effort estimation in the enhanced UCP method

Furthermore, figure 1 visualizes the separation of functional requirements, non functional requirements and project influence. As a consequence, this helps to distinguish between influences on project effort caused by the requirements definition and those caused by the way of project delivery. The interpretation of the UCP terms characterizing the different types of efforts is defined as follows.

2.1 A-Factor

Use cases define the functional requirements related to the scope of the project. In business information systems these requirements are implemented in form of application software (A-Software) [9]. The effort for implementation is proportional to the use case points; it is called A-Factor. To ensure standardized levels of complexity, each use case is rated by the number of its main scenarios, steps and dialogs within the course of a use case.

A **scenario** is defined by its faultless courses achieving the main business goal or alternative goals of the use case. Fault scenarios are only counted if they contain business logic.

A **step** is defined by a self-contained business part of the use case being clearly separated by the adjacent steps, e.g. by change of the actor, by intermediate results or by splitting into scenarios.

A **dialog** is understood as any kind of interface of human interaction and includes e.g. print output. In addition, we also count the usage of interfaces to external systems as dialogs.

A detailed description is provided in the extensive and detailed user guide [5]. We define the following levels of complexity:

- Simple: at most 3 main scenarios, steps and dialogs => 5 points
- Medium: at most 7 main scenarios, steps and dialogs => 10 points
- Complex: 8 or more main scenarios, steps and dialogs => 15 points

The point values {5, 10, 15} are chosen according to the original UCP method [7]. The metric for steps and dialogs is based on large (unpublished) statistical data of the Capgemini group in the field of use cases. We count actors with an increased weight A_j = {5, 10, 15} in analogy to the use cases and in contrast to the original UCP method. If n is the number of use cases, m the number of actors and U_i the point values of the use cases, we define the *A-Factor* in the following way:

$$A - Factor := \sum_{i=1}^{n} U_i + \sum_{j=1}^{m} A_j \tag{1}$$

2.2 T-Factor

It corresponds to the Technical Complexity Factor (TCF) in the original UCP method and covers the non-functional requirements which have influence on effort. In UCP 2.0

Table 1. T-Factor of UCP 2.0

T-Factor ("Technical Factor")		
T_i	**influencing factor**	**weight g_i**
T1	Distributed System	2.0
T2	Performance and load requirements	1.0
T3	Efficiency of the user interface	1.0
T4	Complexity of business rules and calculations	1.0
T5	Reusability	1.0
T6	Easy to install	0.5
T7	Easy to use	0.5
T8	Portability	2.0
T9	Easy to change	1.0
T10	System availability	1.0
T11	Special security features	1.0
T12	Direct access for third parties	1.0
T13	Special user training facilities	1.0

the influencing variables of the T-Factor from the original UCP method are still used but the values of complexity of these variables (0 to 5 points) have been standardized by the definition of examples and analogies. The influencing variables of the T-Factor are presented in table 1.

The standardized descriptions of the T-Factors are not presented in this paper (cf. [6, 8]). Column 3 defines the corresponding weights g_i according to the original method [7]. The T-Factor ranges from 0.58 to 1.28 and is calculated by multiplying the influencing variables T_i with the weights g_i and by summarizing all weighted values:

$$T - Factor := 0.58 + \sum_{i=1}^{13} (T_i \cdot g_i \cdot 0.01) \tag{2}$$

2.3 M-Factor

The management (M-) factor defines the complexity caused by project organization and has been derived from the „Environmental Factor“ (EF) of the original UCP method. The EF has been enhanced to a new advanced M-Factor.

In analogy to the T-Factor, each influencing variable of the M-Factor has been standardized with regard to the level of complexity by definition of examples and analogies. The values again range from 0 to 5 points and the standardizing examples and analogies are not presented in this paper (cf. [6, 8]). Table 2 shows the influencing variables M_i of the M-Factor and their corresponding weights g_i.

Table 2. M-Factor of UCP 2.0

M-Factor ("Environmental Factor")		
M_i	**influencing factor**	**weight g_i**
M1	lead analyst capability	1.4
M2	collaboration (team player)	0.0
M3	personal continuity	0.3
M4	quality of rough specification and T-architecture (architecture/risk resolution)	0.5
M5	process model (process maturity)	1.5
M6	required development schedule	0.0
M7	stable requirements	1.8
M8	number of decision makers	0.0
M9	integration dependency	0.7

The M-Factor ranges from 0.23 to 4.81 and has been standardized to 1.0 (i.e. all factors rated with 3 points). The following formula is used to calculate the M-Factor:

$$M - Factor := \prod_{i=1}^{9} [1 + 0{,}1 \cdot g_i \cdot (3 - M_i)] \tag{3}$$

2.4 Productivity Factor (PF)

The productivity factor (PF) is constant and provides the delivery efficiency rate of the software house. In literature values between 20 and 40 are stated. Due to the higher actor weights, we computed a slightly lower PF of 17.3 h/UCP for our company by calibration with the help of an estimation database containing completed projects.

Both, the M-Factor and the PF are mainly determined by the supplier. The product of all these terms gives the total project effort.

3 Field Study

As described in the introduction the goal of the case study was to find out how big the intrinsic error of the UCP method is and how much the UCP method depends on the format of the specification the estimation is based on. In the following a brief introduction to experiments in the area of software engineering is given followed by the description of the setup of the field study.

3.1 Experimental Setup

The experimental setup in general has to ensure that in a given situation the change X to variable x results in a change Y to variable y [10]. This requires that a change Y being observed is causal determined by X which is the case, if during the experimental setup only X changes.

In the case of effort estimation on different specification formats, it is difficult to ensure that only the specification format changes without changes in the estimation process since the estimator is a human being with learning curves and other influencing factors. This has to be minimized with the means of a controlled experiment setup.

Field study versus laboratory study: In a laboratory study each of the n examined specification formats has to address the same business requirements. In estimation praxis, this is unavailable. Therefore, we focus on a field study with a different project context for each specification format. The most important variables to control for this experiment are:

1. ***Instrumentation:*** Repetition of effort estimations by the same estimator may result in learning curves and therefore interference of the experiment. Therefore, each person creates estimations for (only) two different projects. Furthermore, the same detailed slides and handouts are used for the lectures and exercises at each university.
2. ***Individual differences*** of estimators have been managed by a large scale of estimators and carefully analysis of the experience and the background of these. Each participant (student) attends to a lecture about effort estimation in general and UCP to prepare the exercise (UCP-estimation); mainly information technology students of different universities participated who have the necessary background knowledge on specification of use cases.

3. ***Maturation:*** Effects caused by learning effects during the exercise of this experiment have little to no influence on the results because the participants can easily adjust their results.
4. ***Mortality:*** This means that participants cancel their participation during the field study. For this field study no significant mortality occurred since the estimation was performed in a short period of time (less than 60 minutes) and the participants were highly motivated because of being eager to learn.
5. ***Expectancy and Requirements characteristics:*** Different motivation or attitude to work of the estimators may interfere the field study. To ensure equal task settings for the estimators all instructions were provided literally in the same way.
6. ***Sequence effects*** were controlled by changing the estimation order randomly.
7. ***Sophistication:*** The estimator (reagent) might be influenced by getting a notion of what results are expected by the field study. Therefore, intermediate results have been kept secret during the field study. Further information can be found in [6].

The first step to set up the field study was to find out which types/formats of specifications are used in real industrial projects. Several projects delivered by Capgemini sd&m have been analyzed. Eight formats have been identified (cf. table 3) to be the most common and therefore the most important ones: three of them consist of UML diagrams, four represent textual formats and one uses dialogs/screenshots to describe the functional requirements of the specified system.

Table 3. Identified specification formats (*most common for Capgemini sd&m projects*)

no.	specification formats	description of format
1	activity diagrams	Unified Modeling Language (UML)
2	sequence diagram	Unified Modeling Language (UML)
3	state charts	Unified Modeling Language (UML)
4	rough textual description	textual description
5	tabular description	textual description
6	functional description	textual description
7	business process description	textual description
8	dialog description	uses dialogs/screenshots complemented with textual description

The specifications used in the field study were taken from these real projects and have only been shortened to fit into students exercises time slots and anonymised to satisfy nondisclosure agreements. It was attempted to use each specification format isolated in exactly one specification. In real projects different specification formats are usually mixed up to specify the system functionality. This had to be avoided for the field study to be able to compare the influence of the different formats.

The **next step** was planning and organizing the field study which was executed as a semi controlled experiment. An experiment design was developed: eight specification formats were chosen (cf. table 3); greater 20 results per format have to be gathered; students from eight universities in Germany participated. In [6] it is shown that a

minimum of 20 results per format is a valid and practical sample size to be able to perform a variance analysis using the statistical methods described in chapter 4.1.

3.2 Raw Results of the Field Study Conducted

The following table 4 provides raw results of the field study performed with additional values for mean, median, standard deviation and coefficient of variation for each format.

Table 4. Characteristics of the UCP-distributions, sorted by increasing coefficient of variation *(3. column: „N" = sample size without outliners; 4. column: #UC = number of Use Cases)*

no.	specification format	N	mean #UC	mean [UCP]	median [UCP]	standard-deviation [UCP]	coefficient of variation
4	rough textual description	21	12,9	108,4	106,5	14,9	**0,14**
3	state charts	24	5,3	42,6	43,0	9,4	**0,22**
7	business process description	23	11,5	47,1	45,0	11,3	**0,24**
2	sequence diagram	21	5,3	51,7	50,0	13,2	**0,25**
5	tabular description	29	6,5	70,1	65,0	18,8	**0,27**
8	dialog description	25	8,7	45,7	45,0	12,9	**0,28**
1	activity diagrams	29	13,0	54,7	50,0	15,9	**0,29**
6	functional description	10	6,7	74,6	75,3	23,8	**0,32**

Due to the different mean UCP values the standard deviation cannot be used to compare the different samples (cf. chapter 4.1). Therefore, the coefficient of variation is provided and will later be used to compare these formats. The average dispersion of the UCP values for the different formats (samples) is between 14 % and 32 %. The distributions of the samples and their corresponding formats are visualized using box plots in the following figure 2.

As an additional characteristic to analyze how much the UCP method depends on the format of the specification the number of use cases identified could be used. During the analysis of this field study it was decided that the validity of this characteristic is weak because the variability of the number of use cases is even much higher for most formats than the variability of the UCP results. This is due to the fact that the estimator can e.g. split a medium sized use case in two simple uses cases which does not make a difference for the UCP result but for the number of use cases.

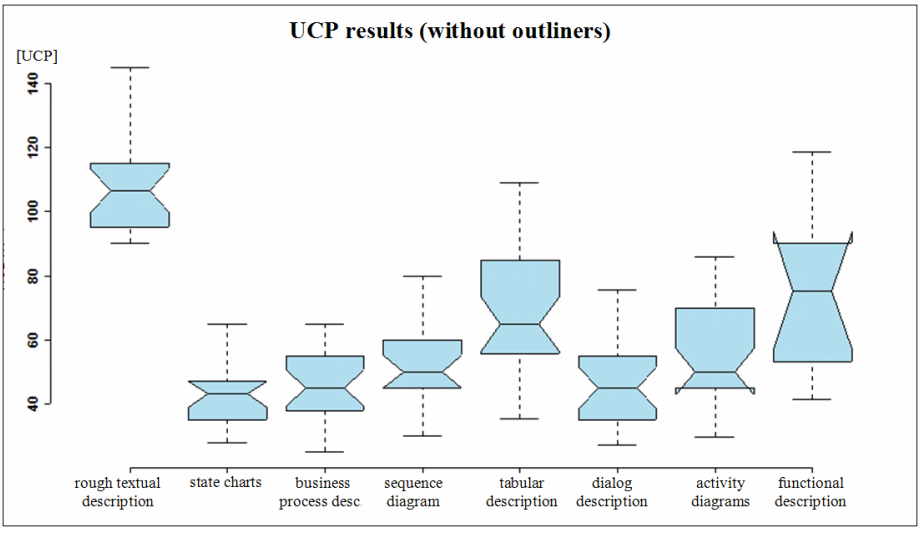


Fig. 2. UCP results visualized using box plots *(results by students, without outliners)*

4 Evaluation of the Field Study

In a next step the statistical evaluation of the experiment outcome has been prepared. We performed a variance analysis to compare the variances of the estimation results corresponding to the different specification formats. Levene´s Test (cf. chapter 4.1.2) was chosen to compare these variances and to cluster formats with similar variance. In the following subchapters the statistical evaluation of the field study is described followed by a comparison of these student results with additionally collected expert results for the same specifications.

4.1 Statistical Methods Used

In the following short paragraphs the statistical methods used to analyse and interpret the data of the field study are described briefly. This should give the reader, who is not familiar with statistics, the required statistical background. Subsequently the transformation of the UCP values and the variance analysis performed are explained.

4.1.1 Coefficient of Variation

Pearson introduced the coefficient of variation V which is a normalized measure of dispersion of a probability distribution [11]. It is defined as the ratio of the standard deviation σ to the mean μ:

$$V = \frac{\sigma}{\mu} \tag{4}$$

The coefficient of variation can therefore be used to compare the variability of samples with different mean values. V is a relative, dimensionless measure of dispersion with the mean of the corresponding sample as its unit. To simplify

interpretation of *V* conversion to a percent-value (multiply by 100) is often used. This describes the variation from the mean value in percent.

4.1.2 Levene´s Test

In statistics, Levene´s test [11] is an inferential statistic used to assess the equality of variance in different samples. It can be used to compare two or more samples at once and is more robust against deviation from normal distribution then the widespread F-Test [11].

Graphically homogeneity of variance can be deduced from the box plots. The boxes and whisker of the samples need to have the same length. Levene´s test is therefore an additional tool to analyze and confirm homogeneity of variances. For two sided problems the pair of hypotheses is as follows:

$$\begin{aligned} H_0 &= \sigma_1 = \sigma_2 = \ldots = \sigma_n \\ H_1 &= \sigma_i \neq \sigma_j \quad \textit{for at least one i, j with } i \neq j \end{aligned} \tag{5}$$

Assuming the 5% level of significance the null hypotheses can be rejected for a *p*-value < 0.05. In that case the alternative hypothesis, which means heterogeneity of variance, has to be accepted. The other way around this procedure cannot be applied because homogeneity of variance does not follow automatically if the *p*-value is greater than 0.05. For a *p*-value close to the value 1.0 together with the observation of the box plots homogeneity of variance can be deduced.

4.1.3 Transformation of UCP Values in Relation to Number of Use Cases

As explained above the coefficient of variation is used to compare the different samples due to the different mean values. Still there is another problem when trying to compare these coefficients of variation. For specifications containing only very few use cases (about less than six), a misjudgment of a single (or a few) use cases will already lead to a quite high deviation from its original total UCP value. From this it follows that the UCP values of the different specifications have to be transformed relative to their mean number of use cases.

For this experiment this means that dispersions of specifications with very few use cases need to be compressed and dispersions of specifications with many use cases need to be stretched out according to the overall mean number of use cases.

Transformation is done by multiplying the UCP value with a constant factor *b* for each sample. The transformation factor *b* is calculated to:

$$b = \frac{\overline{ucCount}}{ucCountMean} \tag{6}$$

where $\overline{ucCount}$ is the mean number of use cases for a specific format and $ucCountMean$ is the overall mean number of use cases. The values of the transformation factor *b* for the samples of this experiment are in-between 0.6 (state diagram) and 1.5 (functional specification). These transformed UCP values [tUCP] allow the comparison of their coefficients of variation *V* and therefore allow comparing the variation of the UCP results for the different specification formats.

Figure 3 shows the box plots of the tUCP values ordered by ascending coefficient of variation.

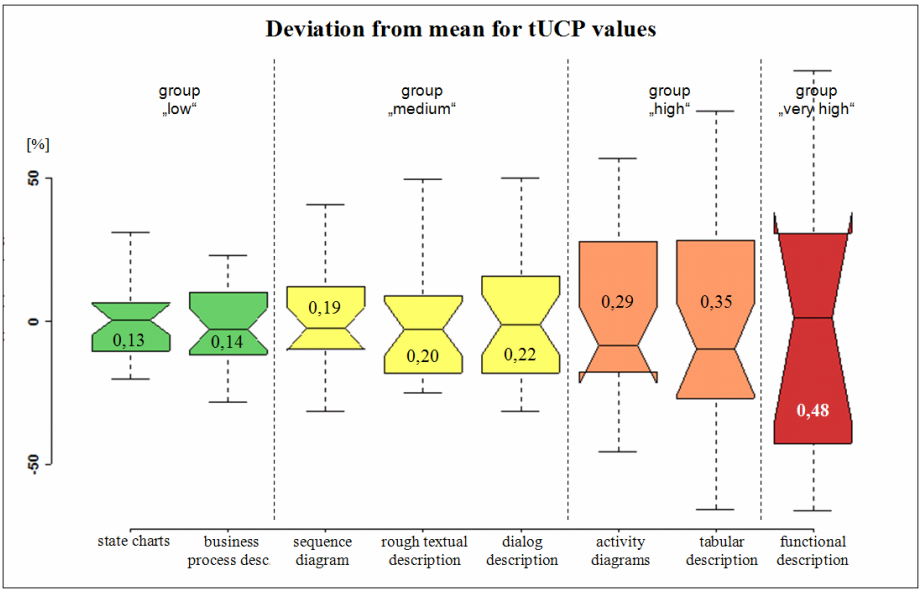


Fig. 3. Deviation from mean for tUCP values in percent *(The explanation of the colors and the group names follows in chapter 4.1.4. The coefficient of variation value for each format is plotted in the middle of each box plot).*

The order of the coefficient of variance values [tUCP] and their corresponding formats do not match with the order of the coefficient of variation values [UCP] (cf. table 4). For example the rough textual description now is only the fourth best format, whereas before the transformation it was the best format. The average number of use cases for this specification is higher than that for most other specifications and therefore the transformation constant *b* is greater than 1. Through normalizing to a comparable number of use cases by the transformation described above the variance of this format was thereby increased.

4.1.4 Variance Analysis Performed

To derive a valuation of the specification formats based on its variability a variance analysis has been performed. For all tests performed the following assumptions are fulfilled:

- Independence of cases - this is a requirement of the design
- The distributions in each of the groups are roughly normally distributed

As shown in table 5 Levene´s test has been applied to several groups of formats. The goal was to identify groups of formats which have approximately the same variability, if UCP estimations are performed by more than one estimator based on the same specification.

Table 5. Groups of specification formats by coefficient of variation *(Groups have been gathered by variation coefficient and proven by Levene´s test: Within one group no variance homogeneity is given whereas between the groups variance homogeneity is given.)*

variability of estimation results	specification format	coefficient of variation	Levene´s test (p-value)	Levene´s test (p-value)
low (13 – 14 %)	state charts	**0,13**	**6.536e^{-06}**	**0,5004**
	business process descript.	**0,14**		
medium (19 – 22 %)	sequence diagram	**0,19**		**0,6592**
	rough textual description	**0,20**		
	dialog description	**0,22**		
high (29 – 35 %)	activity diagrams	**0,29**		**0,3628**
	tabular description	**0,35**		
very high (48 %)	functional description	**0,48**		**-**

The result of Levene´s test applied to all formats (p = 6.536e^{-06}) is highly significant and proofs that variances cannot be equal. Additional intra- and inter-group tests have been performed to find a grouping of the formats. The test results for the intra-group tests are not significant ($p > 0.05$) and therefore the assumption of homogeneity of variances cannot be rejected for the formats of these groups.

The test results for the inter-group tests which are not shown in the table are all significant ($p < 0.05$). Therefore, the assumption of homogeneity of variances between these groups has to be rejected. The variances of these groups differ.

4.2 Comparison with Expert Results

In addition to the field study with students, four experts at Capgemini sd&m were instructed to perform UCP estimations for each specification format based on the same specifications used by the students. The reason was to compare these expert results with the student results of this field study. By this, a rating of the influence of experience could be derived. A comparison with efforts of real projects was not possible due to shortening of the specifications and because of additional project information was withheld. In figure 4 the box plots showing the deviation between expert and student results in percent are visualized.

The formats are sorted by increasing mean deviation. To support the understanding of the diagram the box plot for the format dialog description is chosen to be explained: The box plot shows that 50% of the students achieved UCP results which differ from the experts´ mean UCP value only between -25% and +30%. The median of the students´ results is very close to the experts´ mean value. Therefore, the influence of experience is very low for this format. Even inexperienced estimators can achieve reliable UCP results if estimation is based on that format.

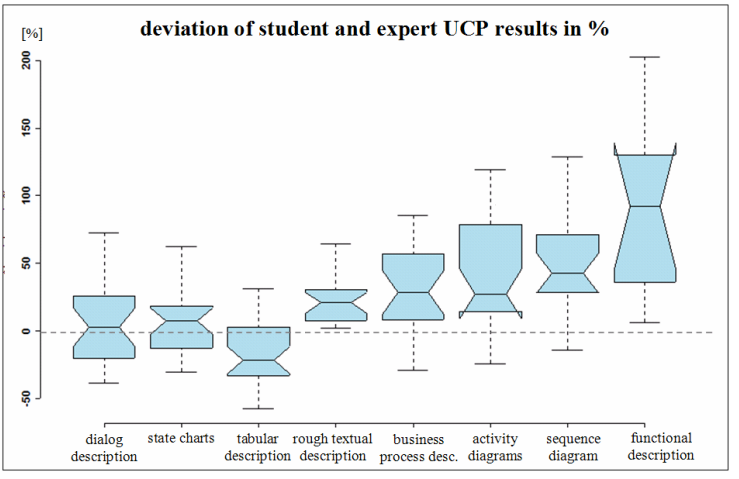


Fig. 4. Deviation of students´ and experts´ UCP results (mean values) in percent

5 Conclusion and Outlook

Within the field study at least 20 estimations by students per format were collected and therefore a valuable statistical evaluation was possible. The mean variance (variation coefficient) of the formats was between 13 and 48% which could be approximately approved by UCP estimations by experts mentioned above.

This field study has proven that the estimation accuracy significantly depends on the underlying specification format. As result, we derived four clusters of the specification formats by variability of the estimation results and we derived a rating of the influence of experience (cf. table 6). A low variability indicates high reproducibility of the UCP estimates. For example, state charts and business process description are very well suited for the UCP method whereas activity diagrams and tabular description result in much less reproducibility.

The clustering by variability of the estimation results was proved with help of Levene´s Test [11]. With the clustering, expert valuation as published in a previous study [5] could now be empirically evaluated and partly confirmed. The rating of the influence of experience was measured by comparing the average use case points from student with results of the four experts. The bigger the distance between average student and expert results, the more influence experience seems to have for the underlying format.

For example, state charts, dialog and tabular descriptions require only low experience by the estimator. On the other hand, the specification with activity diagrams reached a high variability with medium influence of estimators´ experience. This does not correspond to the previous study [5] by experts, who intuitively rated the suitability of the activity diagrams for UCP very good. The reason for this is that steps to be counted for UCP can not always easily be mapped to activities in activity diagrams.

Table 6. Results of field study [6] compared to expert ratings of formats [5]

	expert rating on how good specification format is suitable for UCP-estimates		result field study on suitability of format	
specification format	**expert rating**	**explanatory note to expert rating**	**variability of estimation results**	**influence of experience**
state charts	**average**	**-**	**low (13 – 14 %)**	low
business process description	**average**	depends on degree of detailing		medium
sequence diagram	**good**	if all use-cases are described	**medium (19 – 22 %)**	high
rough textual description	**good**	-		medium
dialog description	**good**	-		low
activity diagrams	**very good**	-	**high (29 – 35 %)**	medium
tabular description	**very good**	-		low
functional description	**good**	depends on comprehension	**very high (48 %)**	high

Another finding is that the intrinsic error of the UCP method decreases with increasing number of use cases. The method should therefore not be used for small projects with less than 50 use cases. Moreover we found, that for some specification formats the UCP method is suited to be applied without extended knowledge. Furthermore, the execution time for performing the UCP estimate does not reveal an impact on the estimation results.

For further investigations it would be necessary to compare the occurred project efforts with the UCP estimations by taking the different specification formats into account. The UCP database at Capgemini sd&m has been extended with the declaration of the specification format to provide more insights in future.

Within this field study eight different projects, based on disparate types of specifications, have been estimated. Thus, only one specification document per specification format type has been analyzed. This might be the source of a systematical error even if much care has been spent on avoiding interfering experimental effects. In further investigations it might be worth while to have different specification formats for identical business requirements (i.e. laboratory study) to approve the presented results.

References

1. Frohnhoff, S., Jung, V., Engels, G.: Use Case Points in der industriellen Praxis. In: Abran, A., et al. (eds.) Applied Software Measurement - Proceedings of the International Workshop on Software Metrics and DASMA Software Metrik Kongress, pp. 511–526. Shaker Verlag (2006)

2. Habela, P., et al.: Adapting Use Case Model for COSMIC-FFP Based Measurement. In: Proceedings of the 15th International Workshop on Software Measurement, Montreal, Canada. Shaker Verlag (2005)
3. Ouwerkerk, J., Abran, A.: An Evaluation of the Design of Use Case Points (UCP). In: Mensura 2006 (International on Software Processand Product Measurement), Cadiz, Spain (2006)
4. Cockburn, A.: Structuring Use Cases with Goals (January 15, 2007), http://alistair.cockburn.us/crystal/articles/sucwg/structuringucswithgoals.htm
5. Frohnhoff, S., Kehler, K., Dumke, R.: Leitfaden zum Finden von Anwendungsfällen für die Use Case Points Methode, Preprint, Fakultät für Informatik, Universität Magdeburg (2007)
6. Engeroff, T.: Analyse der Use-Case-Points-Methode hinsichtlich der zugrunde liegenden Spezifikationsformate, Diplomarbeit, Fachbereich Elektrotechnik und Informationstechnik, Technische Universität Darmstadt (April 2008)
7. Karner, K.: Metrics for Objectory. Diploma thesis, University of Linköping, Sweden, No. LiTHIDA-Ex-9344 (1993)
8. Frohnhoff, S., Engels, G.: Revised Use Case Point Method - Effort Estimation in Development Projects for Business Applications. In: Proceedings of the CONQUEST 2008 - 11th International Conference on Quality Engineering in Software Technology, Potsdam. dpunkt verlag (2008)
9. Siedersleben, J.: Moderne Software-Architektur. dpunkt verlag (2004)
10. Prechelt, L.: Kontrollierte Experimente in der Softwaretechnik: Potenzial und Methode. Springer, Berlin (2001)
11. Sachs, L., Hedderich, J.: Angewandte Statistik – Methodensammlung mit R. Springer, Kiel, 12. Auflage (2006)

Software Measurement @ Siemens – A Practical Approach Allows Best Practice Sharing of Various Organizations

Sebastian Schunk

Siemens AG, Corporate Technology, CT SE SWI,
Otto-Hahn-Ring 6, 81739 Munich, Germany
Sebastian.Schunk@siemens.com

Abstract. The Siemens Measurement System for Software-based Systems (SMS) was set up in 2004 (see Keynote of F. Paulisch at the Metrikon 2006) and until now information of more than 440 projects has been collected. The active contribution of almost all software developing organizations within Siemens to the company-wide Measurement System provides a good data base to compare and analyze performance data such as "Budget Deviation" and the various success-critical influencing factors such as review practices, "Team Size" or "Unplanned Team Changes" between various organizations. This way, areas of interest can be objectively identified, where organizations could benefit from Best Practice Sharing.

Keywords: Measurement System, Benefits, Success Factors, Statistical Analyses, Organization-wide Best Practice Sharing.

1 Motivation

Increasing the performance of software development projects is an area that is of great interest for organizations in the software business as well as the academia. In practice, information on this is drawn either from literature or from benchmarking with other organizations [1]. When comparing performance data such as "Budget Deviation" and "Schedule Deviation" which is set in relation to success-critical influencing factors like review practices, "Team Size" or "Unplanned Team Changes" across different organizations, the analyses often lack trustworthiness due to different underlying definitions. Another major challenge with benchmarking activities between companies is to create mutual trust and openness to share information honestly. Since this key prerequisite is not given in many cases, often the resulting problem is missing comparability of measurement definitions.

Only very few organizations have the possibilities or the willingness to do analyses of their own project development data and draw conclusions about the actual impact of projects' set-ups by internal benchmarking activities.

However, this is beneficial due to various aspects:

- Benchmarking of project performance
- Identification of Best Practice Sharing partners
- Analysis of influencing factors supporting Best Practice Sharing.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 76–85, 2008.

Experiences with the Siemens Measurement System show that the latter aspect is especially valuable in order to maintain the interest and the involvement of the participating organizations. Prior to the detailed explanations of the three aspects, information about the SMS will be given.

2 The Set-Up of the Siemens Measurement System

One of the reasons the Siemens Measurement System for Software-Based Systems (SMS) [2] was set up in October 2004 is to synchronize the measurement activities of the various organizations and to enable them to compare their data on a reliable basis by using the same definitions, etc. The definition of the SMS was based on the international standard ISO/IEC 15939 [3], which is also the basis for the Measurement and Analysis Process Area of the Capability Maturity Model Integration [4]. In addition to the information available within Siemens, external information sources were also considered, including e.g. [5-12].

The SMS encompasses the quantification for several base measures on a metric scale, e.g. "Project duration" or "Project budget", as well as ordinal scaled base measures and nominal characteristics like categories of "Team Changes" or the development model.

The actual data basis is a set of 449 completed software development projects (budget total: EUR 858 mio.) that was collected in a central data repository between March 2006 and May 2008. The projects started between Oct 2004 and Jan 2008 and have at least EUR 100,000 budget with an average of ca. EUR 2 mio. Due to the fact that almost all of the involved organizations contributed actively since March 2006, various analyses on performance and related influencing factors are possible.

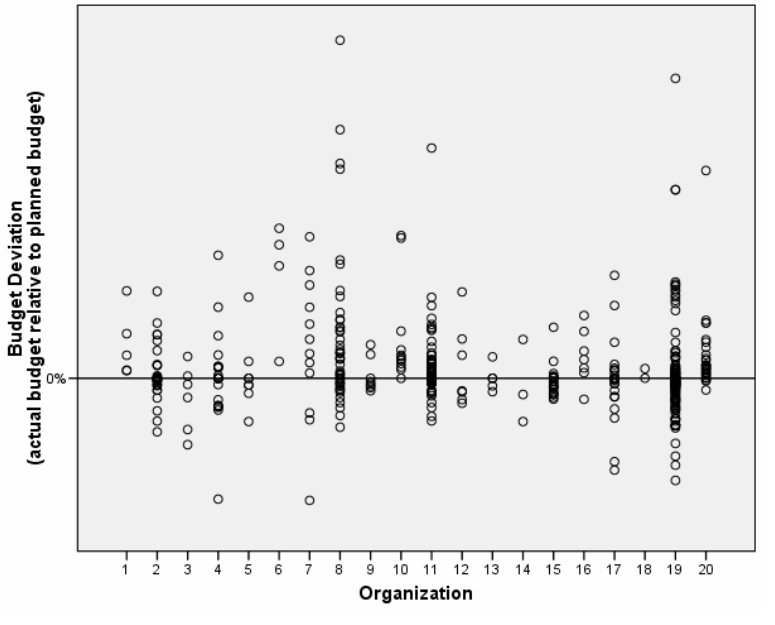


Fig. 1. Budget Deviation of various Organizations

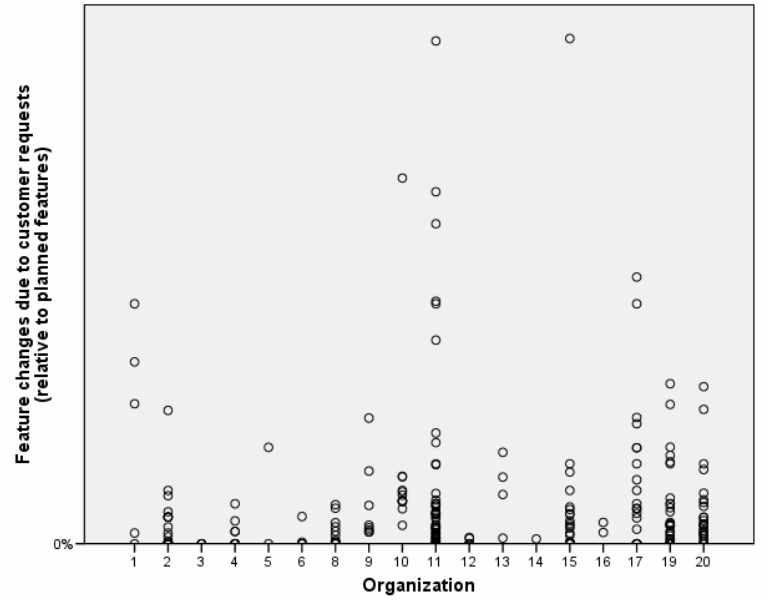


Fig. 2. Feature Changes due to customer requests of various Organizations

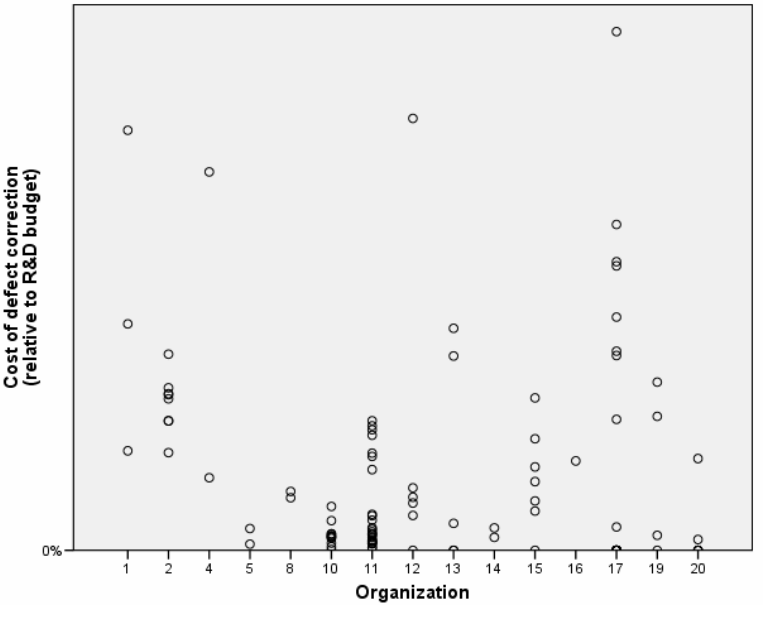


Fig. 3. Cost of Defect Correction of various Organizations

3 Benchmarking of Project Performance

In this paper we present insights of how a measurement system can systematically support Best Practice Sharing-activities across a large organization by offering benchmarking information. To date, one of the major benefits from the SMS is that organizations can use it as source of project performance information. By comparing the projects in terms of e.g. “Budget Deviation” or “Schedule Deviation”, the identification of differences between organizations is possible. This can be seen as an indicator for the actual capability of an organization and provide a basis for process improvement activities.

The weakness of this perspective is that it does not show causes and effects of the variety of influencing factors. This is addressed in section 4.

Between organizations there are significant differences, both regarding the average performance in terms of Budget Deviation and its variance.

There are significant differences between organizations regarding Feature Changes that derive from the customer. Organizations with a constantly low number of Feature Changes might serve as a role model for those which processes do not show stable results.

Costs that arise within 12 month after delivery can be seen as a quality metric. Both the values and the variance differ significantly between organizations.

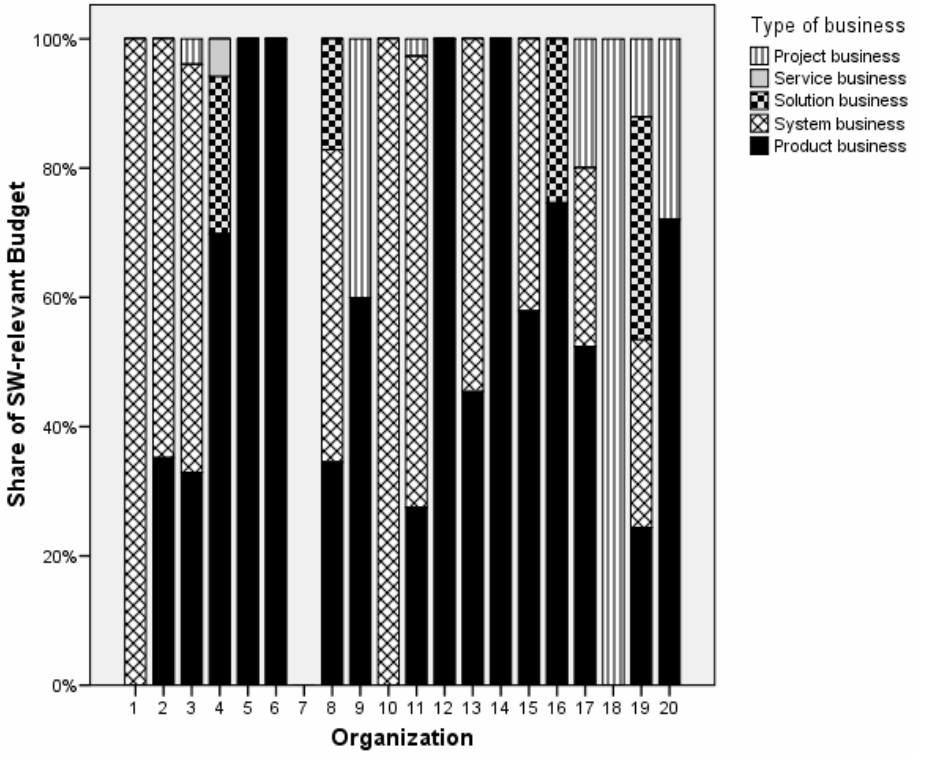


Fig. 4. Type of business of various Organizations

4 Identification of Best Practice Sharing Partners

To be better able to understand causes and effects of the variation of project performance, influencing factors need to be analysed systematically. The SMS allows the identification of organizations where a benchmark is reasonable and Best Practice Sharing is beneficial. Examples for possible categories are "Type of development" (New / Extension / Maintenance), distribution over multiple sites, "Project duration", or development model. Each of the following analyses could be a basis for further Best Practice Sharing.

We will concentrate on two organizations, that show similarities in all of the categories "Type of business", "Type of product" or "Project budget": Organizations "2" and "11" report projects with similar characteristics. This matching pair will be the basis for further analysis presented in section 5.

The most obvious similarity is between organizations "1" and "10", since these organizations only develop software in the "system business". The organizations "2", "3", and "11" are potential benchmarking partners. They show a similar distribution over "product business" and "system business".

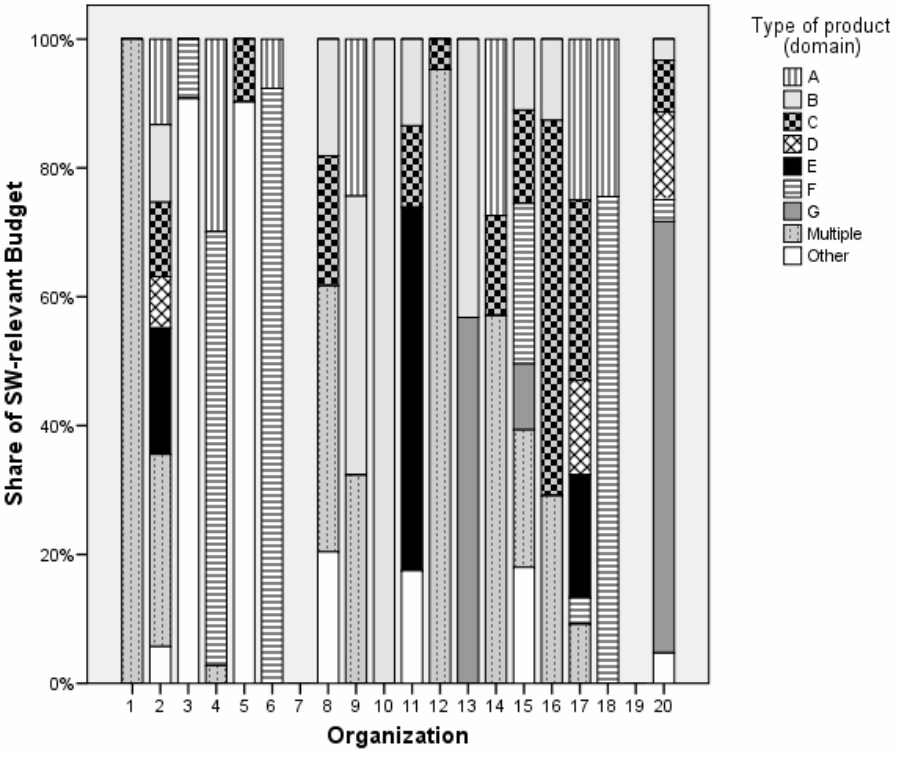


Fig. 5. Type of product of various Organizations

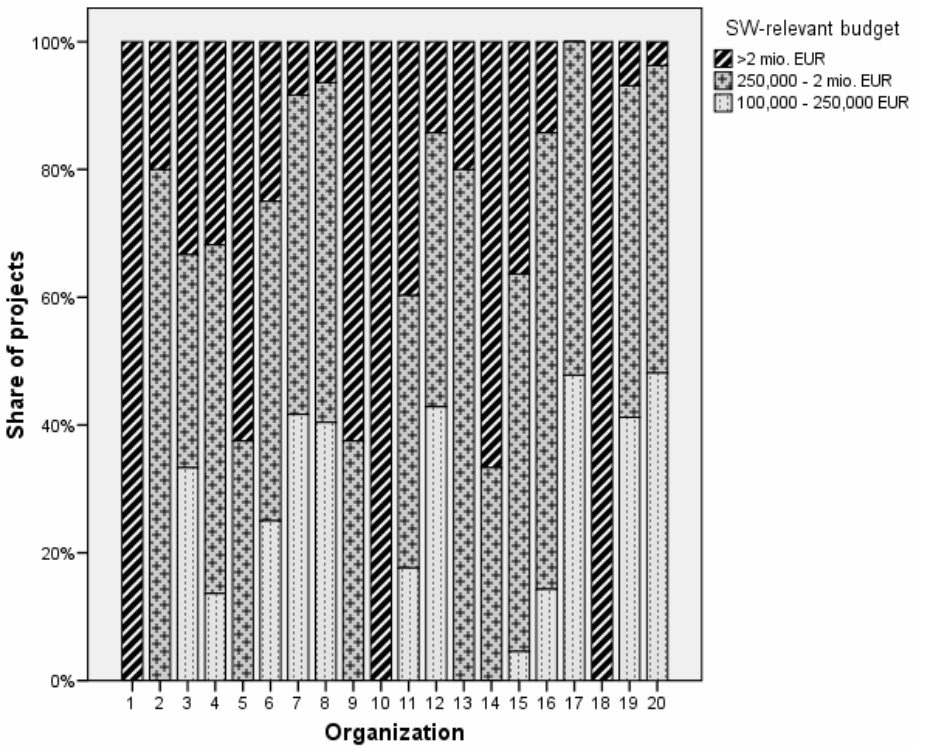


Fig. 6. Project budget of various Organizations

Organizations "2", "11" and "17" are potential benchmark partners as they are the only organizations that develop Type of product "E".

Organizations "1", "10" and "18" develop only projects in the category "large". The organizations "2", "3", "4", "6", "11", "13" and "15", have a similar share of projects in the category "large".

5 Analysis of Influencing Factors Supporting Best Practice Sharing

The major benefit of the SMS is that it provides an organization- and characteristic-specific perspective on the data set which can help improve the steering of both internal processes and projects in practice. In addition to available knowledge that is derived from literature, causes and effects of influencing factors can be observed.

The interest in the analyses and willingness to share information emerges from the organizational setting, especially the same underlying definitions and well-structured reporting and analysis schedule. The analyses can be performed on the basis of all reported projects, single organizations, single aspects like deployed practices and combinations. These insights and, on request, organization specific contact information can be shared,

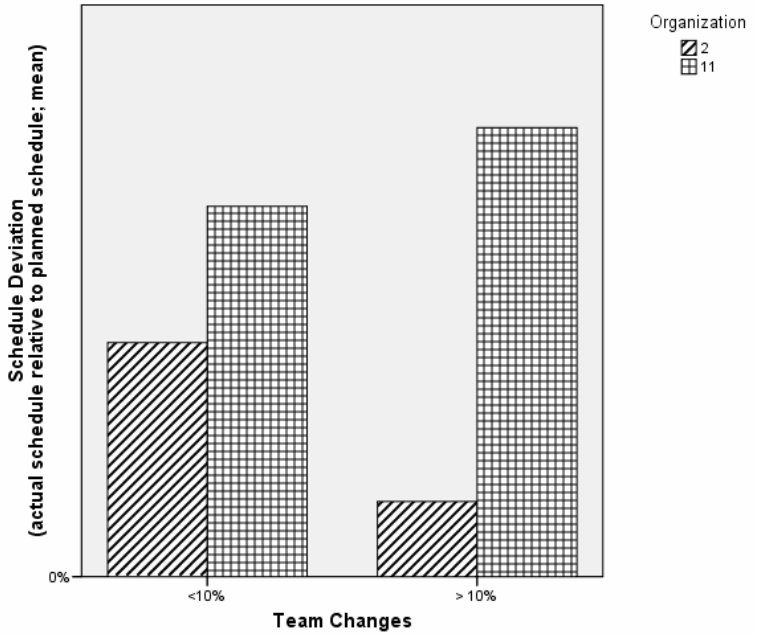


Fig. 7. Unplanned Team Changes vs. Schedule Deviation of selected Organizations

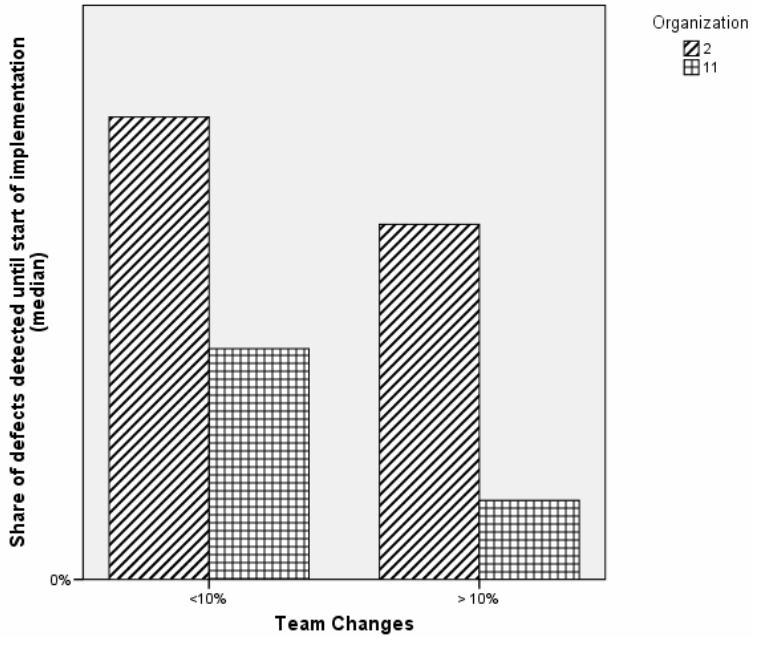


Fig. 8. Unplanned Team Changes vs. Share of defects detected until start of implementation of selected Organizations

enabling e.g. process experts to enter a dialog to exchange experiences on relevant background information of development processes and effort estimation activities. In this paper we focus on the influencing factors. The content and proceedings of further Best Practice Sharing is not in the scope.

The following figures illustrate statistical findings that serve as a basis for Best Practice Sharing of organizations "2" and "11":

In contrast to organization "11", organization "2" is able to complete projects with lower Schedule Deviation even with higher unplanned Team Changes. "11" could benefit from the knowledge and practices of "2".

Analyzing the median of the values, higher values of unplanned Team Changes are strongly related to finding less defects until start of implementation. Both organizations should take this into account.

Analyzing the mean values, the organizations show oppositional trends. The more late defects are found, the more projects exceed schedule at "11". It could benefit from "2" w.r.t. the defect detection.

There is a strong link between the budget and the schedule deviation in organization "11": To improve the Schedule Deviation of large projects, organization "2" could provide helpful information.

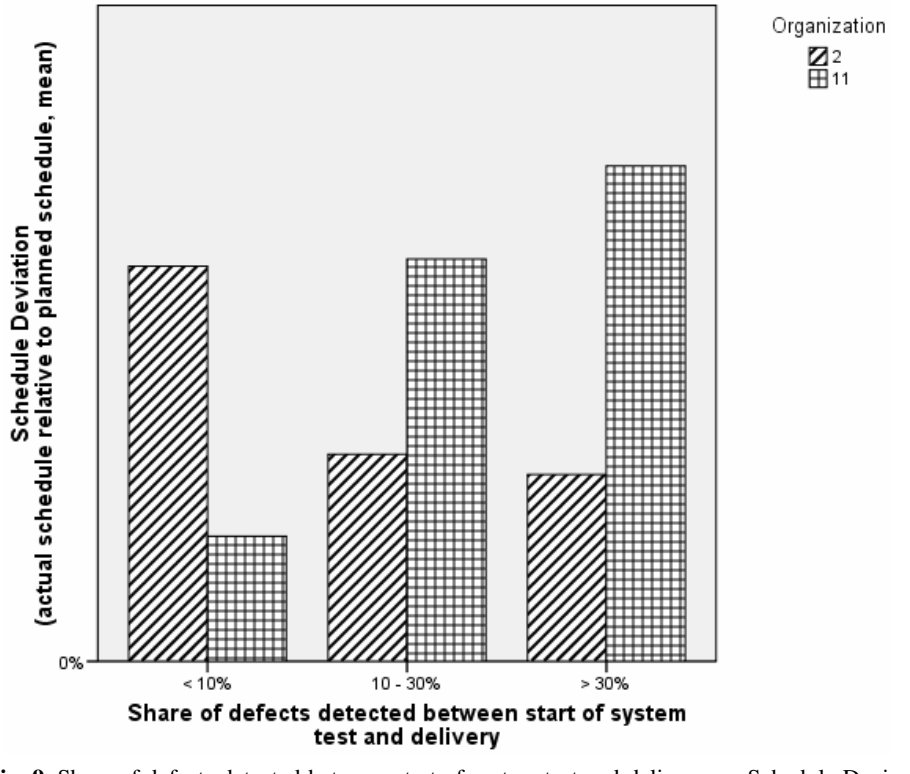


Fig. 9. Share of defects detected between start of system test and delivery vs. Schedule Deviation of selected Organizations

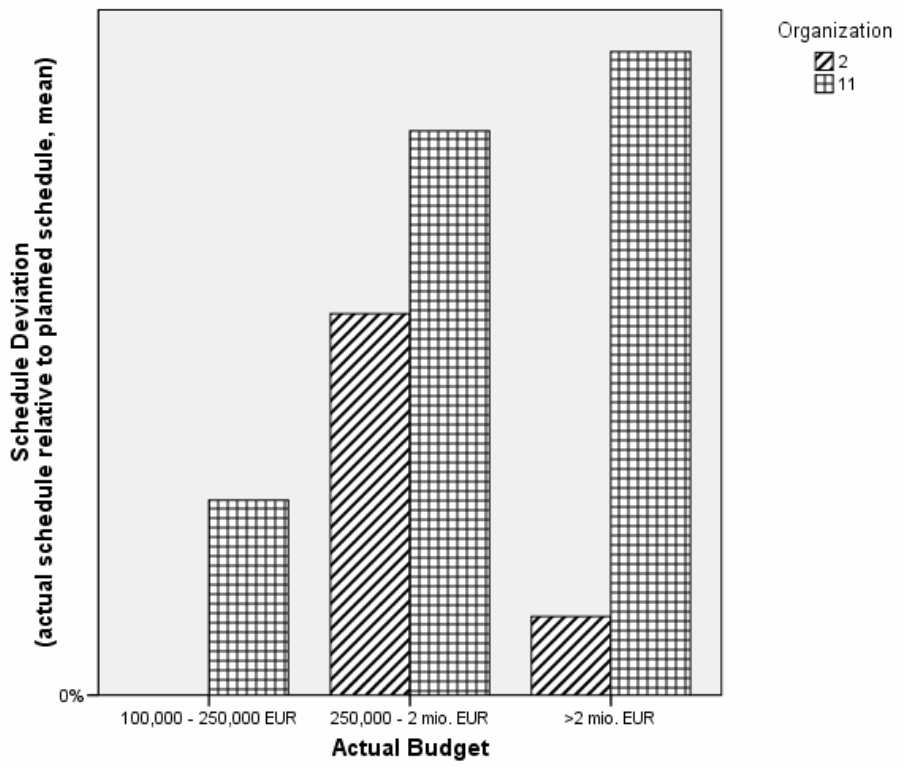


Fig. 10. Actual Budget vs. Schedule Deviation of selected Organizations

6 Lessons Learned

We showed in this paper how a measurement system can support systematic Best Practice Sharing through benchmarking approaches. The background information enables organization representatives or e.g. process experts to compare organizational settings and enter a dialog to exchange experiences and, finally, improve processes.

However, the key precondition to gain valid conclusion from project data is the right amount of transparency and the creation of a common understanding across organizations. This could be reached by involving all organizations in the process of defining the system right from the beginning of the SMS. Making sure that all involved persons of all organizations have the same correct understanding of the definitions, workshops with representatives of all organizations are organized on a regular basis. Non-anonymous benchmarking data is only shared between organizations that agreed on a mutual experience exchange.

Under these circumstances, conclusions based on literature combined with own live data is very convincing. Project leaders or process responsibles can have a higher degree of confidence in the data because they get more background information and also make use of the opportunity to get in direct contact with colleagues of other projects.

References

1. Boehm, B.: Software Cost Estimation with COCOMO II. Prentice Hall, Englewood Cliffs (2000)
2. Paulisch, F.: Establishing a Common Measurement System, In: Applied Software Measurement. In: Proceedings of the International Workshop on Software Metrics and DASMA Software Metrik Kongress, pp. 1–3 (2006)
3. El Emam, K., Card, D. (co-eds.): ISO/IEC Standard 15939: Software Measurement Process, International Organization for Standardization (2002)
4. Crissis, M., Konrad, M., Schrum, C.: Capability Maturity Model – Integration. Addison Wesley, Reading (2003)
5. Briand, L., Differding, C., Rombach, D.: Practical Guidelines for Measurement-Based Process Improvement (Goal Question Metric paradigm), Technical Report (ISERN-96-05) (1996)
6. Ebert, C., Dumke, R., Bundschuh, M., Schmietendorf, A.: Best Practices in Software Measurement. Springer, Heidelberg (2005)
7. McGarry, J., Card, D., et al.: Practical Software Measurement. Addison Wesley, Reading (2002)
8. Goethert, B., Bailey, E.K., Busby, M.B.: Software Effort & Schedule Measurement. SEI, Carnegie Mellon University (1992)
9. Grady, R.B.: Practical Software Metrics for Project Management and Process Improvement. PTR Prentice-Hall, Englewood Cliffs (1992)
10. Herbsleb, J.D., Grinter, R.E.: Conceptual simplicity meets organizational complexity: case study of a corporate metrics program. In: Proceedings of the 20th international conference on Software engineering, Kyoto, Japan (1998)
11. Pfleeger, S.L., Jeffery, R., Curtis, B., Kitchenham, B.: Status report on software measurement. Software, IEEE 14(2) (March/April 1997)
12. Wiegers, K.E.: Software Metrics: Ten Traps to Avoid. Software Development magazine (October 1997)

Measurement Support for Effective Supplier Management

Christof Ebert*

Vector Consulting Services, Ingersheimerstr. 24, 70499 Stuttgart, Germany
christof.ebert@vector-consulting.de

Abstract. Many companies work with suppliers to either increase flexibility, focus on their own core competencies or reduce cost. Often, after a while into such externalized engineering or supply activities they realize that savings are much smaller and problems are more difficult to cure than before. Others realize that suppliers do not deliver according to initial commitments. This article provides experiences and empirical evidence from global software engineering and supplier management over several years in different context and companies.

Keywords: Supplier management, global software engineering, risk management, process improvement.

1 Introduction

Many companies work with suppliers to either increase flexibility, focus on their own core competencies or reduce cost [2]. An increasing share also takes advantage of skill availability in different parts of the world or quality awareness and process focus in some specialized industries and suppliers. Most of these latter companies engage globally active outsourcing companies to achieve fastest ramp-up of their sourcing and globalization targets.

After a while into that business they realize that savings are much smaller and problems are more difficult to cure than before. Others realize that suppliers do not deliver according to initial commitments.

What went wrong? Why do so many companies struggle to achieve the targets they initially set for their supplier projects? What can we do better to make distributed or multi-party software engineering a success? These questions have stimulated this article where we integrated experiences and empirical evidence from global software engineering and supplier management over several years in different context and companies.

With an increasing amount of software components, services, outsourcing and resources acquired from various external suppliers, professional supplier management is a core activity of any project manager and controller. Such distributed software

* Some parts of this article appeared first in Ebert, C., and R.Dumke: Software Measurement. Copyrights, Springer, Heidelberg, New York, 2007 [1]. Used with permission. We recommend reading that book as an extension to the concepts and benchmarks mentioned in this article.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 86–96, 2008.

development poses substantial risks to project and product management. Not all eventualities can however be buffered, because in the global economy, developing and implementing products must be fast, cost effective and adaptive to changing needs. Therefore, there is a need to utilize different techniques to effectively and efficiently mitigate risks.

This article introduces supplier management from the perspective of measurement and controlling activities that ensure visibility throughout a project. We highlight the top-ten supplier related risks as we have identified them over the past decade in a multitude of projects and situations covering four continents. They are not specific to an industry or company size, but rather to the underlying life-cycle processes and management practices. Based on concrete industry experiences and enhanced with consulting work in several other globally acting companies, we show impacts of these risks and also how they can be effectively mitigated by proper supplier management. Our focus is on measurement techniques to improve supplier management.

We will look to sourcing management from a generic perspective. We have worked in different industries on a variety of different sourcing models, be it specific component development and evolution, COTS acquisition, outsourced services or global software engineering.

For this paper we have studied the project results over years and analyzed the correlation between risk mitigation actions and project deliveries. This paper is a result of empirical study of historical project data together. We have verified the validity of risk mitigation actions by establishing a strong correlation with expected end results of projects. Our experience clearly demonstrated that applying the discussed risk mitigation methods, has resulted in improved delivery schedules [1,2].

2 Risks with Sourcing of Software and Services

To systematically identify software and service sourcing related risks and evaluate appropriate risk mitigation, we will look to the major drivers for sourcing and externalization and then elaborate how they are impacted by respective risks. Throughout our research over the past 10 years on global software engineering, we see four major drivers fueling the need for external sourcing and acquisition, namely efficiency, presence, talent and flexibility. Fig. 1 provides an overview on these drivers and how they relate to specific sourcing risks. Let us look to these four sourcing drivers and related risks. Risk management will then be illustrated with concrete benchmarks in the following chapter.

1. Efficiency. Software and IT companies need to deliver fast and reliably while at the same time the competition is literally a mouse click away. Hardly any other business has so low entry barriers as IT and therefore stimulates an endless fight for efficiency along the dimensions of improved cost, quality and time to profit. GSE (global software engineering) and software sourcing clearly helps in improving efficiency due to labor cost differences across the world, better quality with many well-trained and process-minded engineers especially in Asia and shorter time to profit with following the sun and developing and maintaining software in two to three shifts in different time zones. Directly related risks to the efficiency target are project delivery failures and insufficient quality.

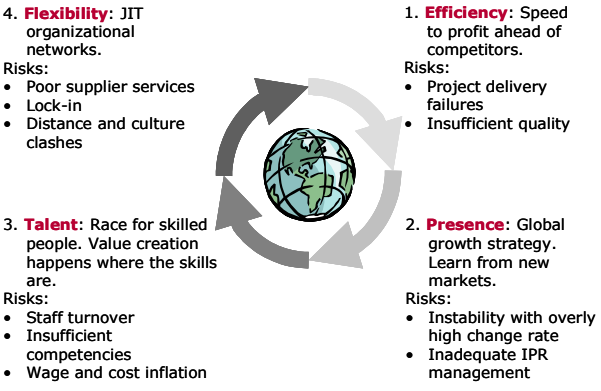


Fig. 1. GSE drivers and related risks

2. Presence. Distributed and externalized R&D and software engineering has become part of companies' growth strategies, because they are closer to their markets and they better understand how to cope with regional needs, be it software development or services. Such global growth is a self-sustaining force, as it demands increasing capacities in captive or outsourced software engineering centers. Directly related risks to the ambition of presence in distributed markets are instability with overly high change rates (requirement changes) and inadequate IPR management. Risk of requirement changes is specifically included here, as we observed higher rate of change in distributed teams when compared to co-located teams.

3. Talent. Computer science and engineering skills are scarce. Many countries do not have enough resources locally available to cope with the demand for IT and software products and services. Fueling this trend, many younger people got nervous with media-driven misperceptions about the danger of sourcing for the entire field of software, that they decided to rather engage in fully different fields. The consequence is a global race for excellent software engineers. Sourcing is the instrument to provide such skills and handle the related supplier-processes. Directly related risks to the drive for global talent allocation are staff turnover rates, insufficient competences and wage and cost inflation.

4. Flexibility. Software organizations are driven by fast changing demands on skills and sheer numbers of engineers. With the development of a new and innovative product many people are needed with broad experiences, while when arriving in maintenance, these skill needs look different and manpower distributions are also changing. Such flexible demand can not anymore be handled inside the enterprise. Sourcing is the answer to provide skilled engineers just in time and thus allows building flexible eco systems combining suppliers, customers with engineering and service providers. Directly related risks to the flexibility goal are poor supplier services, lock-in, and distance & culture clashes.

Obviously not all companies that engage in software and service sourcing look to all four drivers with the same motivation. As a matter of fact, we even see a kind of trajectory where a vast majority of companies starts with efficiency needs (i.e., cost

savings), and then moves on to presence in local markets, and only after these two forces are understood moves further to talent and flexibility. Also, it is clear that these four factors feed themselves. The more energy a company spends on for instance building a regional pool of skilled software engineers, the more it also considers how to best utilize these competencies to, for instance, build a regional market or develop new products for such markets. In consequence not all companies will face mentioned risks in same depth and at the same time.

Depending on the specific sourcing-layout (e.g., with or without external supplier), the ranking list of these top-ten risks is as follows:

1. Project delivery failures
2. Insufficient quality
3. Distance and culture clashes
4. Staff turnover (mostly for captive centers)
5. Poor supplier services
6. Instability with overly high change rate
7. Insufficient competences
8. Wage and cost inflation
9. Lock-in
10. Inadequate IPR management

In order to validate this risk list, we performed two types of analysis. First we did a profound analysis of sourcing projects during the timeframe of 1996 to 2007. Looking to over hundred projects performed either by sourcing suppliers for componenhts and COTS-software or in software centers in India, China, Brazil and Eastern Europe, we found a consistent pattern of the top ten risks. We then validated this initial list by looking to companies we are consulting with and also to published field studies [2,3,5]. Depending on the specific sourcing-layout (e.g., with or without external supplier), the ranking list of these top-ten risks is as follows:

3 Mitigating the Risks: Effective Supplier Management

Increasingly software and related services are sourced from a multitude of suppliers. The entire supply chain has changed, and only few companies still try to master all different layers of their product (or service) architectures [3]. It is hard to find software that is genuinely developed by a single company at one place, except start-up situations but even then, they typically source components from partners or use open source software. Mostly it is components that are sourced from external suppliers. But often it is engineers that are sourced into the development, such as outsourcing or global engineering teams. Basic concepts of global software engineering and lessons learned are detailed and discussed in [2].

Working with an external supplier that provides the engineering services in an offshore (or nearshore) scenario is shown in Fig. 2. The supplier of the (global) engineering services will from day one build strong interfaces to the major functions in the product life-cycle of its client. Again, these interfaces are shown with arrows. Interface management is the clear professional need of the supplier; while for the client it looks certainly like overhead. And frankly it is overhead, but born out of risk

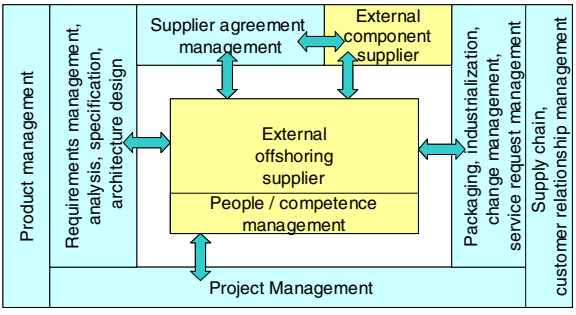


Fig. 2. Global software engineering with external supplier

mitigation of the supplier who otherwise would fear that changes would continuously ripple from the client to his organization making it impossible to keep service level agreements (SLA) and delivery commitments. Such interfaces cost an additional 10-20 % (depending on the maturity of both client and supplier) on top of regular project cost without any value-add as seen from the client [1,3]. With several other suppliers involved for component deliveries such overheads can grow into the 30 % range. Needless to say, people management and competence management are handled inside the supplier (not shown for the external component supplier which is even more separated) on the basis of forecasts delivered by means of the contract and regular client stakeholder reviews.

Sourcing projects fail if tasks are broken down too much, such as asking a remote engineer doing the verification for software develop concurrently in another site. Here distance effects and lack of direct communication slow down development rather than helping it. The single biggest source of difficulties in any acquisition project is related to communication across sites, bad communication hindering both coordination and insufficient management processes. For instance, continuous integration of insufficiently verified and encapsulated software components fails if done remote to the parallel ongoing software development. Distributed teams working on exactly the same topic (e.g., the famous follow-the-sun pattern of developing a piece of software in different time zones) pose highest challenges for coordination and often resulted in severe overheads that would be measurable or tangible only later (e.g., features misinterpreted, insufficient quality, lack of ownership and responsibility, and so on).

Fig. 3 shows the relevant phases along the product life-cycle and the respective activities related to supplier management. Four major phases are distinguished, namely supplier strategy, supplier selection, contract management, and relationship management. The figure shows typical work products that must be available at these phases.

Initially an individual client or customer organization must provide a realistic and precise expectation of functional and nonfunctional requirements (e.g., reliability). They should clearly state that payment will be provided only for systems that meet the agreed upon functionality (e.g., requirements, acceptance tests, SLA conditions). They should demand require milestone presentations of progress for continued funding.

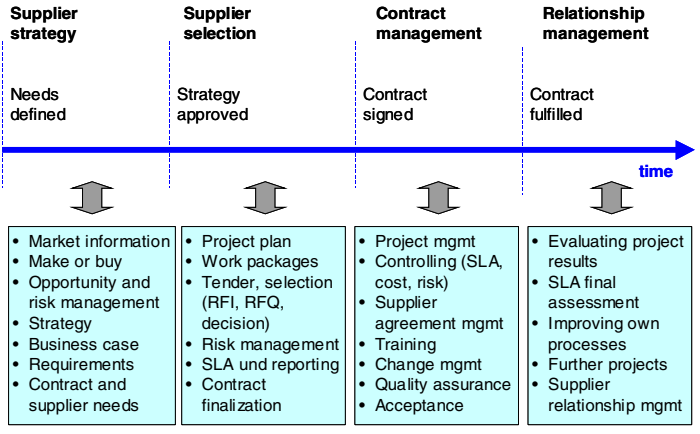


Fig. 3. Supplier agreement management across the entire sourcing cycle

Some simple checks for supplier selection should be applied throughout the different processes of supplier selection and agreement management:

- Did you ever work with this supplier and would you do it again? What were the lessons learned from that previous contract? Alternatively demand this check from a reference client who you know and trust.
- What expertise and references are available from the supplier in your own domain?
- What is the turnover rate at the supplier site? Is it acceptable or rather high? How are skills managed in light of this turnover rate? What turnover rates will be assured by the contract?
- How stable is the supplier and its management or shareholders? Did it recently change, reorganize or merge with another company? Avoid any supplier that is currently hampered by big acquisitions.
- What business processes are in place at the supplier to elicit requirements and to cope with change? Does this fit your needs?
- Is the supplier able and experienced in handling global development teams? Can it manage teams with members from different companies?
- Does the supplier and its employees have the necessary formal qualifications your customers and markets demand (e.g., ISO 9001, CMMI maturity levels, etc.)? Is the supplier periodically audited? Check some recent audit results.
- Are the legal constraints acceptable for you and your company? Often suppliers demand that the site for legal disputes be in a part of the world where you are not so experienced. Check which site makes sense for you and your lawyers. Check if there are some sample legal cases that show typical behaviors. Specifically focus on anything related to protecting your intellectual property. Manage upfront the risk of any impact on your intellectual properties; such as if a key engineer would defect.
- Is the infrastructure sufficient for your own purposes? Does it scale up to the high interaction needs during shared development or testing? Is it protected and

auditable? Are the tools interfaces to your own tools sufficient? Have they been tested in real-world scenarios before?
- What prices are demanded for the services? Are they competitive? How will you avoid a lock-in position once the supplier has understood your technology, products and business?

Generally speaking there are many checks which should be performed prior to the contract signature and determine a first "go / no-go" for the selection. Most can be done offline as part of a request for quotation. You might still want to visit the suppliers' sites to directly see offices and talk with engineers or management. In that case assure you speak with those engineers and team leads working later on your project. Trust your feeling when looking into offices or cafeterias. They provide lots of messages about culture and behaviors.

Supplier organizations on the other hand must insist on a signed contract with requirements. They must agree before contract execution on clear and reasonable acceptance criteria. The contract must be explicit that the supplier owns the software until final payment. They must clearly agree on liabilities and support after handover. They have to express disagreement and unrealistic conditions openly and not continue with diverging assumptions. They should always strive for win-win results and therefore offer compromise approaches, once needs are understood. In case of component delivery they should include a software key that will operate after the date of contracted software acceptance.

Supplier monitoring is done similarly to what we described in the chapter on global software engineering and sourcing, namely on project and on contract levels. Responsibilities might be split in your company so that the project manager observes the contract execution in his own project and scope (e.g., deliverables, cost control, quality levels, schedule), while a procurement and sourcing manager holds responsible for the overall contract execution and observing that conditions from the frame contract are fulfilled.

As a contractor (for a supplier, independent whether it is services or specific component delivery) you should always consider the golden rule of supplier management: You pay for what you get. Do not get trapped into contracts that look "cheap" and later bring tons of extra cost due to lousy processes and insufficient delivery quality. Preconditions of any successful supplier management are good processes on both sides, i.e., for the client and the supplier. Insufficient client processes cannot be externalized. They will not scale up from a single site to several sites. Often those low-maturity processes can be handled in localized development without many overheads due to collocated teams, but will fail with globalization.

In fact, process maturity – on both sides, namely client and supplier – matters when it comes to successful supplier management [1,2,3]. The SEI has build upon these experiences and developed not only the CMMI for development [4] (useful from a perspective of engineering systems) but also the CMMI for acquisition [5] which is highly useful to set up and improve the supplier management related processes, such as supplier selection or supplier evaluation.

If your own processes (as a client) are on a CMMI maturity level 1 or 2 you better ask for a consultant who can help you in installing effective engineering and management processes. Most suppliers offer such support, but this is not necessarily a

sustainable solution, as they have different interest and business models. Independent how your processes look like, it is relevant to review them carefully with your suppliers and agree interfaces on work product, engineering and tools level. The exchange of information must be carefully planned. A change management tool is not enough. It needs rules for documentation, design reviews, change management boards, and so on. Install workflow management and online accessible project, work product and process information to ensure proper knowledge management. Interactive process models, such as RUP and others have proven very helpful to communicate and install processes.

As a supplier you should strive for high maturity, for several reasons. The market attractiveness in software-driven industries is extremely high, due to low entry barriers and continuous push for innovative products. You are in strong global competition for excellence. Suppliers have recognized that better process maturity ensures better schedule and SLA performance, productivity and quality. Why should not they demand it along their supply chain. High market attractiveness continues to push to cost reduction and efficiency improvement. Many new entries start each day with similar business ideas that drive your own company. If you do not continuously improve both products and processes, they will do it. The best example is the move of high-technology products to Asia, as there is a much higher competitive pressure for high process maturity than in North America or Europe.

Fig. 4 summarizes the dependencies between supplier and client process maturity. The win-win situation within a supply chain is driven by moving to high maturity in both dimensions. A low maturity supplier will eventually be replaced for the reasons given before. A low maturity client working with a high-maturity supplier will face extra cost and overheads and thus try to move upwards at fast pace. This move is fueled by the client's own market pressure from competitors that are already optimizing their product development processes. The upper left and lower right quadrants therefore are no stable plateau but will always create forces and momentum towards the upper right quadrant – thus sustaining a win-win relationship between supplier and client.

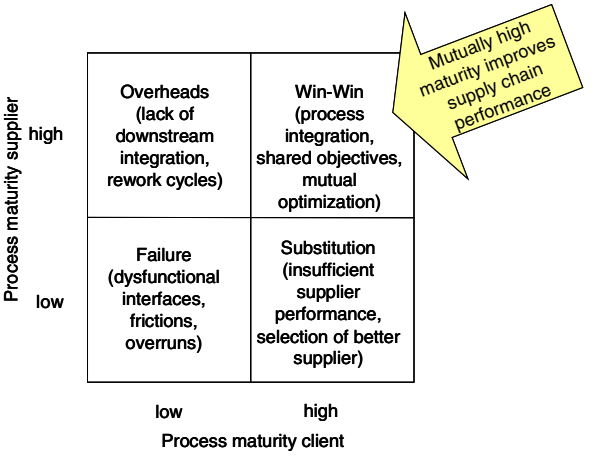


Fig. 4. The supplier-client relationship benefits from high maturity on both sides

4 Measurements for Supplier Management

Monitoring cost, progress and performance of global software projects is a control activity concerned with identifying, measuring, accumulating, analyzing and interpreting project information for planning and tracking activities, decision-making, and cost accounting. Project (or supplier) monitoring and control is the basic tool for gaining insight into project performance and is more than only ensuring the overall technical correctness of a project. Global projects or supplier sourcing is typically done with two different methods for the project level and for the contract level.

Project level: Project monitoring and tracking is done with techniques such as earned value, budget adherence, schedule adherence, project reviews, and so on. Traditional project tracking looked to actual results against plans, where the plans would be adjusted after the facts indicate that they are not reachable. This method creates too many delays and is not sufficiently precise to drive concrete corrective actions on the spot. For global development projects, such monitoring often means that difficulties accumulate too long. Therefore, continuous predictions should be used to relate actual constraints and performance to historical performance results. Good forecasts allow adjusting plans and mitigating risks long before the actual performance tracking measurements would visualize such results. For instance, knowing about average mean time to defect allows planning for maintenance staff, help desk and support centers, or service level agreements.

Contract level: Contracts are absolutely crucial for managing external suppliers. The must include Service Level Agreements (SLA) with defined targets, threshold values, and so on for deliverables, schedules, reaction time of services, quality levels and expenses. Quality criteria are defined by outages, downtime, number of service requests, residual defects in the subsequent phase or by phase-end or hand-over criteria. These targets must be measurable. Independently of the global collaboration model or contract model is established with suppliers, it is key to set the right targets and set them as performance indicators for R&D the management in each location. SLAs are set up at the beginning of a contract (or even for a long-term frame contract) and controlled continuously on the supplier side and periodically on the buyer side. For internally hosted GSE it might be wise (depending on organization structure) to govern by means of internal contracts and SLAs. They have the big advantage that targets and measurements are agreed upfront and would not need continuous debates with senior management if some delivery is late. Certainly such internal contracts and SLAs together with a culture of accountability and clearly assigned responsibility also avoid the political game of finger pointing to “the others” that did not do their job well.

A word on work allocation and ownership. Shifting coherent activities (e.g., design and verification of related work products) to low-cost countries is highly inefficient. Often tasks are overly fragmented and the quality control activities are handled with poor results due to lack of knowledge. In the end each delivery has to be checked twice, at the time it is shipped to a low-cost country and then again backwards. All this costs time and money – and it demotivates engineers on both sides, as it always ends up in ping-pong. As said before, we strongly recommend building teams preferably in one place and assigning them ownership for a work product including functionality and quality. Such teams should operate globally according to needs and

skills availability, but not be internally split into first and second-class engineering tasks.

Global software engineering and sourcing need the traditional project monitoring activities combined with contract level supervision. Measurements include the following dimensions:

- Project launch: feasibility analysis, requirements quality, process capability (own and that of external or offshore organizations).
- Results against plan: earned value, budget adherence, schedule adherence, global development teams performance, site productivity.
- Quality: detected defects per activity, residual defects, reliability forecasts.
- Forecasts: cost to complete, time to complete.
- Supplier contracts: service level agreements, cost evolution, productivity and quality of suppliers, performance evolution, supplier rating.

These different dimensions should be considered when setting up contracts and SLAs (be it internally with distributed development or externally with suppliers).

5 Conclusions

Effective supplier management is very much depending on the measurement capabilities of an organization. Too often suppliers are believed to deliver according to specifications and SLA, and suddenly the client realizes that this was mere wishful thinking. Let us therefore conclude with concrete supplier measurement-oriented hints:

- Set up clear and measurable service level agreements. Ensure that this SLA contains all that matters for you in the contract.
- Insist on periodic reporting according to the SLA.
- From the beginning define thresholds that establish when and how insufficient performance will be escalated.
- Measure supplier capability or demand such measurement based on industry standards, such as CMMI for development and for acquisition.
- As a supplier or customer move towards high maturity product development by using the CMMI and its maturity level 3-5 concepts of process excellence and quantitative management. Do not stagnate on maturity levels 1-3 as you will be eventually replaced.
- Relate value you receive from suppliers to the risk and cost of the delivered services or components. Manage the risk of lock-in and dependencies that could create extra risk and cost.
- Implement contract evaluation after each single project. Go beyond the qualitative checklist and report into measurements and fact-based lessons-learned.

The projected benefits of software and service sourcing must be carefully balanced with additional cost that might occur only at a later point. This includes loosing ownership; cost overheads related to traveling, relocation, communication or middle management or redundant development and test equipment; unavailability of

dedicated tools for distributed work environments; impacts of the learning curve that slows down with more locations involved; cultural differences which can impact work climate; insufficient language skills; different legal constraints related to work-time, organization, or participation of unions; building up redundant skills and resource buffers to be prepared for co-located teams and for unforeseen maintenance activities; and not the least morale erosion of own staff when the supplier network is getting bigger, while the own vision might fade. We therefore strongly recommend to first build and agree a clear sourcing strategy, then engage into specific supplier selection and management processes, and along the various sourcing activities closely monitoring value, risks and problems versus initial expectations.

Author

Christof Ebert is managing director at Vector Consulting Services. He is helping clients worldwide to improve product development. Prior to that, he held world-wide engineering and management positions at Alcatel for more than a decade. A senior member of IEEE, Dr. Ebert authored several books and serves as a frequent keynote speaker at conferences. Contact him at christof.ebert@vector-consulting.de.

References

1. Ebert, C., Dumke, R.: Software Measurement. Copyrights. Springer, Heidelberg (2007)
2. Meyers, B.C., Oberndorf, P.: Managing Software Acquisition: Open Systems and COTS Products. Addison-Wesley, Reading (2001)
3. Ebert, C.: Global Software Engineering. IEEE ReadyNote (e-Book). IEEE Computer Society, Los Alamitos (2006) (Cited 09. September 2008), http://www.computer.org/portal/cms_docs_cs/ieeecs/jsp/ReadyNotes/displayRNCatalog.jsp
4. CMMI® for Development, Version 1.2. CMU/SEI-2006-TR-008 (August 2006)
5. CMMI® for Acquisition, Version 1.2. CMU/SEI-2007-TR-017 (November 2007)

Measuring Distances for Ontology-Based Systems

Steffen Mencke[1], Cornelius Wille[2], and Reiner Dumke[1]

[1] Otto-von-Guericke University of Magdeburg
paper@mencke-online.com,
dumke@ivs.cs.uni-magdeburg.de
[2] University of Applied Sciences Bingen
wille@fh-bingen.de

Abstract. Nowadays, measurement and assessment of artifacts within the area of software development are of high concern for scientific institutions. With the increasing usage of the Web 2.0, together with service-oriented applications, the importance of Web-based systems is still growing. Ontologies as a fundamental concept of the Semantic Web as envisioned by Tim Berners-Lee, play an important role for current and future applications.

To promote the high flexibility of this technology for international dynamics, additional concepts for the Semantic Web are necessary. Ontology metrics are used to measure certain aspects of ontologies. In this paper, a set of novel distance metrics is introduced and their applicability is proven by the presentation of an example.

Keywords: Semantic Web, Ontology, Metric.

1 Introduction

Semantics (from the Greek word *semantikos* = significant) in general is the meaning of something or more specifically the study of meaning [1]. Often, additional information is needed to shift from information processing to knowledge processing. Those semantic annotations provide the technological basis for many advanced applications.

Ontologies are one key technology of the Semantic Web, as envisioned by Berners-Lee [2]. Together with explicit representations of the semantics of data for machine-accessibility, such domain theories are the basis for intelligent next generation applications for the Web and other areas of interest [3] with a special focus on knowledge sharing and reuse. Ontologies are defined as a specification of a conceptualization [4]. Or in other words: they are the formal representation of an abstract view of the world. They define for example: a vocabulary for the unambiguous meaning [5], a taxonomy for classification of entities [3] and a content theory, (due to the definition of classes of objects, relations and concept hierarchies) [5]. Furthermore, they enable consistency checking [3].

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 97–106, 2008.

Top-level application areas identified by [6] are collaboration, interoperability, education and modelling. Application domains are not limited, too. Ontologies are useful, wherever semantic information can enhance certain tools, products or processes (e.g. e-Learning ([3], [7]), Virtual Engineering [8]).

2 Distance-Based Semantic Windows

For certain use cases the concept of a 'Semantic Window' was defined in [9] and [10]. This term describes a set of elements of a given ontology within a certain multi-dimensional distance around a focus element. Dimensions for its definition are related to the concepts of an ontology as well as to the datatype properties. Furthermore, instances and taxonomic as well as non-taxonomic relations can be taken into consideration.

For the further detailing, the description starts with a specialized redefinition of an ontology $O = (C, R, D, I)$, where C is the set of ontological concepts following a taxonomic structure, $R = R_{tax} \cup R_{ntax}$ is the set of object properties/relations taxonomically and non-taxonomically relating two concepts $R_{ij}(C_i, C_j)$ and D is the set of datatype properties/attributes of the ontology. I is the set of instances. An ontological component of each of these types can be the enrichment point for the Semantic Window. From this, four different aspects, the dimensions of the Semantic Window, can be derived.

- Concept view
- Datatype property view
- Object property view
- Instance view

For each of the four views, distance measures are defined for the existing dimensions. A help function is $f^{niv}(C_i)$ describing the level of the concept according to its taxonomic level with $f^{niv}(C_{root}) = 0$ (Formula 1). Function $f^{parent}(C_i, C_j)$ delivers back the first more abstract concept shared by C_i and C_j, if it exists and is connected to them only via $R \in R_{tax}$(Formula 2). $f^{tax}(C_i, C_j)$ (Formula 3) and $f^{ntax}(C_i, C_j)$ (Formula 4) determine the length of the taxonomic or non-taxonomic path of object properties from C_i to C_j (the result is -1, if there does not exist such a path).

$$f^{niv} : \text{Concept} \mapsto \text{Integer}. \tag{1}$$

$$f^{parent} : \langle \text{Concept, Concept} \rangle \mapsto \text{Concept}. \tag{2}$$

$$f^{tax} : \langle \text{Concept, Concept} \rangle \mapsto \text{Integer}. \tag{3}$$

$$f^{ntax} : \langle \text{Concept, Concept} \rangle \mapsto \text{Integer}. \tag{4}$$

f^{tax} and f^{ntax} can be realised as described in the equations 5 and 6.

$$f^{tax}(C_i, C_j) = \begin{cases} 0 & \text{if } C_i \equiv C_j, \\ 1 & \text{if } |f^{niv}(C_i) - f^{niv}(C_j)| = 1 \\ & \wedge R_{ij}(C_i, C_j) \in R_{tax}, \\ f^{tax}(C_i, C_k) + 1 & \text{if } f^{tax}(C_i, C_k) = n \wedge f^{tax}(C_k, C_j) = 1, \\ -1 & \text{otherwise.} \end{cases} \tag{5}$$

$$f^{ntax}(C_i, C_j) = \begin{cases} 0 & \text{if } C_j \equiv C_j, \\ 1 & \text{if } R_{ij}(C_i, C_j) \in R_{ntax}, \\ f^{ntax}(C_i, C_k) + 1 & \text{if } f^{ntax}(C_i, C_k) = n \wedge f^{ntax}(C_k, C_j) = 1, \\ -1 & \text{otherwise.} \end{cases} \tag{6}$$

In the following, the dimensions of the concept view are described. The rest of the formulas are described in [9].

2.1 Concept Dimensions from the Concept View

The dimensions of the distance related to the ontology's concepts having a concept as the focusing point are defined in equations 7 to 10. The single distance measures relate to the abstraction dimension distance c^{abs}, to the specialization dimension distance c^{spec}, to the sibling dimension distance c^{sib} and to the non-taxonomic dimension distance c^{ntax}. They measure the distance between the focusing point concept C_F and another concept C_j of the ontology.

$$c^{abs}(C_F, C_j) = f^{niv}(C_F) - f^{niv}(C_j). \tag{7}$$

$$c^{spec}(C_F, C_j) = f^{niv}(C_j) - f^{niv}(C_F). \tag{8}$$

$$c^{sib}(C_F, C_j) = f^{niv}(C_F) - f^{niv}(f^{parent}(C_F, C_j)). \tag{9}$$

$$c^{ntax}(C_F, C_j) = f^{ntax}(C_F, C_j). \tag{10}$$

The equations above are restricted by: $C_F, C_i, C_j \in C$. Equation 7 is restricted by: $f^{niv}(C_F) > f^{niv}(C_j)$ and $f^{tax}(C_F, C_j) \neq -1$. Equation 8 is restricted by: $f^{niv}(C_F) < f^{niv}(C_j)$ and $f^{tax}(C_F, C_j) \neq -1$. Equation 9 is restricted by: $f^{niv}(C_F) = f^{niv}(C_j)$ and $f^{niv}(f^{parent}(C_F, C_j)) < f^{niv}(C_F)$.

2.2 Datatype Property Dimensions from the Concept View

The dimensions of the distance related to the ontology's datatype properties having a concept as the focusing point are defined in equations 11 to 14. The

single distance measures relate to the abstraction dimension distance d^{abs}, to the specialization dimension distance d^{spec}, to the sibling dimension distance d^{sib} and to the non-taxonomic dimension distance d^{ntax}. They measure the distance between the focusing point concept C_F and a datatype property D_j of the ontology. $C(D_j)$ is the concept that a datatype property D_j belongs to.

$$d^{abs}(C_F, D_j) = f^{niv}(C_F) - f^{niv}(C(D_j)). \tag{11}$$

$$d^{spec}(C_F, D_j) = f^{niv}(C(D_j)) - f^{niv}(C_F). \tag{12}$$

$$d^{sib}(C_F, D_j) = f^{niv}(C_F) - f^{niv}(f^{parent}(C_F, C(D_j))). \tag{13}$$

$$d^{ntax}(C_F, D_j) = f^{ntax}(C_F, C(D_j)). \tag{14}$$

The equations above are restricted by: $C_F, C_i, C_j, C(D_j) \in C$ and $D_j \in D$. Equation 11 is restricted by: $f^{niv}(C_F) > f^{niv}(C(D_j))$ and $f^{tax}(C_F, C(D_j)) \neq -1$. Equation 12 is restricted by: $f^{niv}(C_F) < f^{niv}(C(D_j))$ and $f^{tax}(C_F, C(D_j)) \neq -1$. Equation 13 is restricted by: $f^{niv}(C_F) = f^{niv}(C(D_j))$ and $f^{niv}(f^{parent}(C_F, C(D_j))) < f^{niv}(C_F)$.

2.3 Object Property Dimensions from the Concept View

The dimensions of the distance related to the ontology's object properties having a concept as the focusing point are defined in equations 15 to 19. The single distance measures relate to the abstraction dimension distance r^{abs}, to the specialization dimension distance r^{spec} as well as to the (abstraction and specialization) sibling dimension distance $r^{sib^{abs}}$ and $r^{sib^{spec}}$. The non-taxonomic dimension distance is measured by r^{ntax}. They measure the distance between the focusing point concept C_F and an object property R_j of the ontology.

$$r^{abs}(C_F, R_j) = \begin{cases} 0 & \text{if } \nexists R_j(C_F, C_j), \\ 1 & \text{if } \exists R_j = R_{Fj}(C_F, C_j) \\ & \wedge f^{niv}(C_F) > f^{niv}(C_j), \\ r^{abs}(C_F, R_i) + 1 & \text{if } r^{abs}(C_F, R_i) = n \\ & \wedge c^{abs}(C_i, C_j) = 1 \\ & \wedge R_j = R_{ij}(C_i, C_j) \\ & \wedge R_i = R_{hi}(C_h, C_i) \\ & \wedge f^{niv}(C_h) > f^{niv}(C_i) \\ & > f^{niv}(C_j). \end{cases} \tag{15}$$

$$r^{spec}(C_F, R_j) = \begin{cases} 0 & \text{if } \nexists R_j(C_F, C_j), \\ 1 & \exists R_j = R_{Fj}(C_F, C_j) \\ & \wedge f^{niv}(C_F) < f^{niv}(C_j), \\ r^{spec}(C_F, R_i) + 1 & \text{if } r^{spec}(C_F, R_i) = n \\ & \wedge c^{spec}(C_i, C_j) = 1 \\ & \wedge R_j = R_{ij}(C_i, C_j) \\ & \wedge R_i = R_{hi}(C_h, C_i) \\ & \wedge f^{niv}(C_h) < f^{niv}(C_i) \\ & < f^{niv}(C_j). \end{cases} \tag{16}$$

$$\begin{aligned} r^{sib^{abs}}(C_F, R_j) = c^{sib}(C_F, C_h) | c^{abs}(C_h, C_i) = 1 \\ \wedge R_j = R_{ij}(C_i, C_j) \\ \wedge f^{tax}(C_h, C_i) \neq -1 \\ \wedge f^{niv}(C_h) > f^{niv}(C_i) > f^{niv}(C_j). \end{aligned} \tag{17}$$

$$\begin{aligned} r^{sib^{spec}}(C_F, R_j) = c^{sib}(C_F, C_h) | c^{spec}(C_h, C_i) = 1 \\ \wedge R_j = R_{ij}(C_i, C_j) \\ \wedge f^{tax}(C_h, C_i) \neq -1 \\ \wedge f^{niv}(C_h) < f^{niv}(C_i) < f^{niv}(C_j). \end{aligned} \tag{18}$$

$$r^{ntax}(C_F, R_j) = \begin{cases} 0 & \text{if } \nexists R_j(C_F, C_j), \\ 1 & \text{if } c^{ntax}(C_F, C_j) = 1 \\ & \wedge \exists R_j(C_F, C_j), \\ r^{ntax}(C_F, R_i) + 1 & \text{if } \exists r^{ntax}(C_F, R_i) = \\ & f^{ntax}(C_F, C_i) = n \\ & \wedge R_i = R_{hi}(C_h, C_i) \\ & \wedge R_j = R_{ij}(C_i, C_j). \end{cases} \tag{19}$$

The equations above are restricted by $R_j, R_{Fj}, R_{ij} \in R$ and $C_F, C_h, C_i, C_j \in C$. Equations 15 to 18 are further restricted by $R_{Fj}, R_i, R_j \in R_{tax}$. For equation 19 the following restrictions apply: $R_i, R_j \in R_{ntax}$.

2.4 Instance Dimensions from the Concept View

The dimensions of the distance related to the ontology's instances having a concept as the focusing point are defined in equations 20 to 23. The single distance measures relate to the abstraction dimension distance i^{abs}, to the specialization dimension distance i^{spec} and to the sibling dimension distance i^{sib} as well as the non-taxonomic dimension distance is measured by i^{ntax}. They measure the distance between the focusing point concept C_F and an instance I_j of the ontology. $C(I_j)$ is the concept that an instance I_j is instantiated of.

$$i^{abs}(C_F, I_j) = f^{niv}(C_F) - f^{niv}(C(I_j)). \tag{20}$$

$$i^{spec}(C_F, I_j) = f^{niv}(C(I_j)) - f^{niv}(C_F). \tag{21}$$

$$i^{sib}(C_F, I_j) = f^{niv}(C_F) - f^{niv}(f^{parent}(C_F, C(I_j))). \tag{22}$$

$$i^{ntax}(C_F, I_j) = f^{ntax}(C_F, C(I_j)). \tag{23}$$

The equations above are restricted by: $C_F, C_i, C_j, C(I_j) \in C$ and $I_j \in I$. Equation 20 is restricted by: $f^{niv}(C_F) > f^{niv}(C(I_j))$ and $f^{tax}(C_F, C(I_j)) \neq -1$. Equation 21 is restricted by: $f^{niv}(C_F) < f^{niv}(C(I_j))$ and $f^{tax}(C_F, C(I_j)) \neq -1$. Equation 22 is restricted by: $f^{niv}(C_F) = f^{niv}(C(I_j))$ and $f^{niv}(f^{parent}(C_F, C(I_j))) < f^{niv}(C_F)$.

Every distance measure described above delivers back -1, if the function's arguments are not appropriate according to the nature of the distance to be measured.

2.5 Size of the Semantic Window

Within a Semantic Window, from any ontological element's point of view, all distances as well as the ontological element being the focusing point are given and used to determine a set of ontological elements W containing all ontological elements those distance are smaller than the given ones. The distances are summarized in vectors as demonstrated below.

In the following, an example is sketched to show the usage of distances to determine a Semantic Window. A graphical representation of the result is shown in Figure 1. Filled circles represent concepts, filled squares represent instances and filled diamonds represent datatype properties, all being located within the Semantic Window.

The focusing point is a concept and the concept distances are given in vector 24, datatype property distances in vector 25, object property distances in vector 26 and the instance distances are given in vector 27.

$$dist^C(C_6) = \begin{pmatrix} c^{abs} \\ c^{spec} \\ c^{sib} \\ c^{ntax} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{24}$$

$$dist^D(C_6) = \begin{pmatrix} d^{abs} \\ d^{spec} \\ d^{sib} \\ d^{ntax} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{25}$$

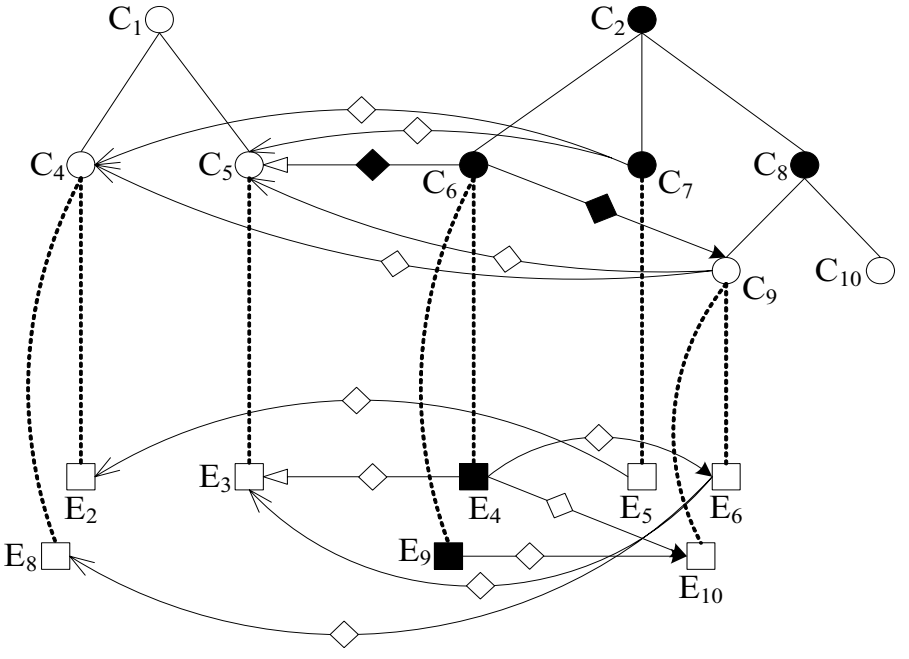


Fig. 1. Example for a Distance-Based Semantic Window with C_6 as Focusing Point and the Defined Distances in Vectors 24 to 27

$$dist^R(C_6) = \begin{pmatrix} r^{abs} \\ r^{spec} \\ r^{sib^{abs}} \\ r^{sib^{spec}} \\ r^{ntax} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{pmatrix} \tag{26}$$

$$dist^I(C_6) = \begin{pmatrix} i^{abs} \\ i^{spec} \\ i^{sib} \\ i^{ntax} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{27}$$

3 Ontology-Based Content Enrichment in e-Learning Systems

E-Learning-related content is any portion of data that can be displayed to a user by the runtime part of an e-Learning system. According to this, content enrichment describes the process of searching and displaying additional information, being semantically related to the information of the e-Learning content. The work is also valuably usable for other users of e-Learning systems, for example content creators, learning unit authors or didactical experts.

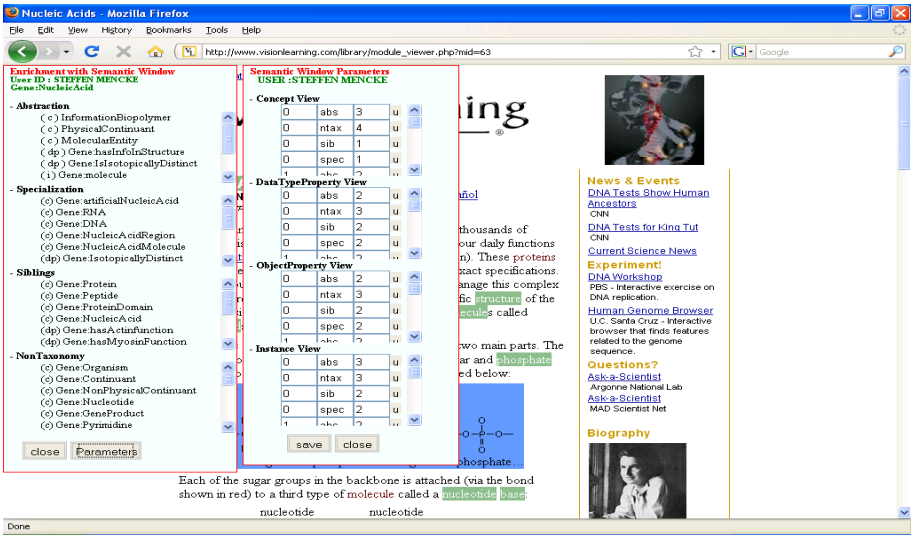


Fig. 2. Distance-Based Semantic Windows for Content Enrichment

A tool was implemented based on the presented distance metrics for ontologies. It follows the algorithms introduced in [10]. Figure 2 presents a screenshot of this tool. Following a given domain ontology it identifies appropriate focus elements and highlights them. They are the starting point for content enrichment. Predefined distance measures are used to determine the size of the Semantic Window – to identify the subset of the ontology that includes additional valuable information. A user-based adaption of the dimensions as well as the sizes of the Semantic Window is possible.

With this tool, the improvement of enrichment is possible, too (see Figure 3). Users can add new enrichment content in order to complete the available data sources and thereby to collaborate on quality improvement.

By the definition of distances between ontology components other improvements become possible, too.

A possible example is presented in [9]. There, those metrics are used to determine the distance between didactical approaches make assumptions about their similarity. The lower distance between two concepts in a hierarchical representation of existing approaches is, the more similar the didactical approaches are.

The balance of ontologies is another usage area for ontology distance metrics. In [11] certain starting points are presented. It is suggested, that all distances from a center of the ontology to the boundary should be similar – then an ontology can be balanced.

The algorithms presented in [12] also can use distance metrics for ontologies. Specific, semantically based learning types can be identified by checking the

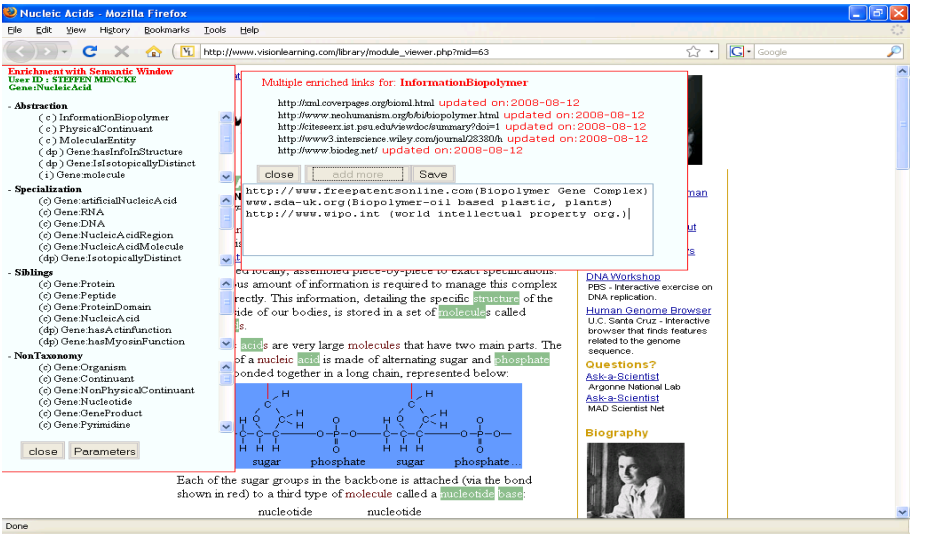


Fig. 3. Enriched Web Page Based on Semantic Windows

type of ontological distances between learning steps from one learning object to another one. After that, new learning objects can be suggested to the learner. By this, a more adapted course presentation is possible.

4 Conclusion and Further Work

In this paper, distance-based metrics for ontology were presented. Furthermore, their advantages were proved by the presentation of selected use cases from the e-Learning domain. Especially the enrichment of given resources with additional content was targeted – a prototype was presented and explained.

Although implementation examples show their usefulness, ontology distance metrics can be further improved. Possible approaches are definition of transitive Semantic Windows or the introduction of additional weights.

References

1. Encyclopedia Britannica - Online: Semantics. Link (2008), http://www.britannica.com/eb/article-9110293/semantics
2. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American 284(5), 34–44 (2001)
3. Devedžić, V.: Semantic Web and Education. Springer's Integrated Series in Information Systems. Springer, Heidelberg (2006)
4. Gruber, T.R.: A Translation Approach to Portable Ontology Specifications. Knowledge Acquisition 5(2), 199–220 (1993)
5. Chandrasekaran, B., Josephson, J.R., Benjamins, V.R.: What Are Ontologies, and Why Do We Need Them? IEEE Intelligent Systems 14(1), 20–26 (1999)

6. Fikes, R.: Multi-Use Ontologies. Stanford University (1998) (Retrieved February 07, 2007), http://www-ksl.stanford.edu/people/fikes/cs222/1998/Ontologies/sld001.htm
7. Mencke, S., Dumke, R.: A Hierarchy of Ontologies for Didactics-Enhanced E-learning. In: Proceedings of the International Conference on Interactive Computer aided Learning (ICL 2007), Villach, Austria (2007)
8. Vornholt, S., Mencke, S.: Ontologies for the Virtual Engineering Process. In: Proceedings of the 11. IFF-Wissenschaftstage, Magdeburg, Germany (2008)
9. Mencke, S.: Proactive Ontology-Based Content Provision in the Context of e-Learning. Ph.D thesis, Otto-von-Guericke University of Magdeburg (2008)
10. Mencke, S., Rud, D., Zbrog, F., Dumke, R.: Proactive Autonomous Resource Enrichment for e-Learning. In: Proceedings of the 4th International Conference on Web Information Systems and Technologies (WEBIST 2008), Funchal, Madeira, Portugal, vol. 1, pp. 464–467. INSTICC Press (2008)
11. Mencke, S., Kunz, M., Dumke, R.: Towards Metrics for Ontology Balance. In: Proceedings of the Twentieth International Conference on Software Engineering and Knowledge Engineering (SEKE 2008), Redwood City, USA (2008)
12. Mencke, S., Kunz, M., Zenker, N., Dumke, R.: Ontology-Based Generic Learning Path Recommendations. In: Proceedings of the 2008 International Conference on e-Learning, e-Business, Enterprise Information Systems, and e-Government (EEE 2008), Las Vegas, Nevada, USA (2008)

Challenges in Evaluating SOA Test Processes

Ayaz Farooq, Konstantina Georgieva, and Reiner R. Dumke

Institute for Distributed Systems, University of Magdeburg,
P.O. Box 4120, 39106 Magdeburg, Germany
{farooq,georgieva,dumke}@ivs.cs.uni-magdeburg.de

Abstract. Service-oriented architecture enables creation of enterprise-wide and cross-enterprise flexible, dynamic business processes and agile applications. The performance, reliability and other quality aspects of such systems, thus, become very important for the success of businesses. Testing is a way to evaluate these quality attributes. However, the new unique architecture style of SOA-based systems calls for reorienting testing procedures, methods, techniques, and tools etc. In this paper we discuss how we can evaluate efficiency and effectiveness of SOA test process. Considering an existing generic test process evaluation framework, we attempt to highlight areas where necessary adjustments to this framework are needed to care for the specialized testing perspectives of SOA-based systems.

Keywords: Service-oriented architecture, software test process, test process evaluation, test process improvement, software measurement.

1 Introduction

Service-oriented computing represents a new generation distributed computing platform [1]. A service is a primary building block of systems built this way and the resulting software programs as services can be discovered, composed, instantiated, and executed at runtime. This new service-oriented architecture (SOA) style of software system structure is aimed at improving efficiency, agility, and productivity of the enterprises. An SOA is believed to allow flexibility, better alignment with business goals, and cost-effectiveness of the developed systems.

The rigor and flexibility of SOA-based systems comes with a price and confronts us with unique challenges [2] [3]. Some example research issues in this context are business related (SOA strategy selection etc.), engineering (process and lifecycle, development, quality assurance, testing, and maintenance etc.), operations (service monitoring and support etc.), and cross-cutting issues (governance and stakeholder management etc.) [4].

One issue within the above mentioned engineering related research areas is quality assurance and testing. Software testing itself is a complex task and its scope and objectives vary with the applicable software engineering dimensions such as technology (object-oriented, component-based, services-based etc.), development methodology (waterfall, agile, etc.), and application systems (information systems, embedded systems etc.). The widespread adoption of SOA-based

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 107–113, 2008.

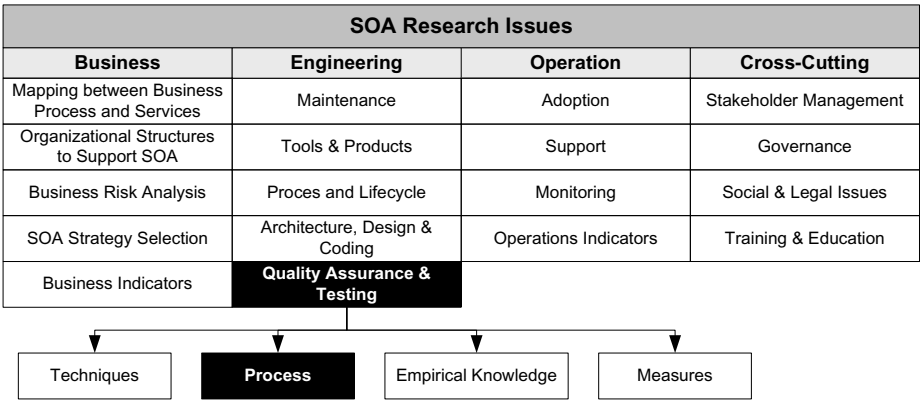


Fig. 1. SOA Research Directions

solutions introduces rising concern for efficient and effective testing methods but unfortunately testing SOA has not yet received adequate attention [5] [6].

Like traditional software testing, a significant portion of an SOA-based development project resources is consumed by testing phases. Since optimization of development effort, time, and cost has always been prime concern of software companies, the choice eventually falls on analyzing testing processes for their efficiency and effectiveness. Figure 1 summarizes various research issues within SOA world. A meta-measurement framework has recently been proposed to evaluate such and few other quality dimensions of the testing processes [7]. Our paper investigates challenges and issues in adopting this generic evaluation framework in the context of SOA testing processes.

In the remaining part of this paper, section 2 summarizes various testing aspects of SOA systems. Section 3 introduces the generic test process evaluation framework and its major components. Challenges to implementing this approach for SOA are discussed in section 4. The paper concludes in section 5 identifying future research work in this direction.

2 Testing of SOA Systems

The new architecture style of SOA-based systems brings about its own testing challenges [8] [9]. Furthermore, many of the existing testing methods, techniques, and tools cannot directly work with SOA. For example, lack of a user interface and unavailability of access to service code will affect the unit testing techniques. Specific testing concerns relevant to service-oriented systems vary with service quality attributes (performance, security, reliability, inter-operability, vulnerability etc.), testing levels (unit, integration, functional, non-functional, regression etc.), and testing perspectives (developer, provider, integrator, certifier, user etc.). Consequently, in comparison with conventional software development

projects, SOA demands even a larger fraction of the development effort to be dedicated to testing and quality assurance.

Within the context of testing and quality assurance of SOA-based systems, we have found that focus of research has primarily been on developing new testing techniques, maturity models, measures, and establishing a testing process. We have surveyed several articles, conference proceedings, and books and found 127 references relating to SOA testing. Results of our preliminary analysis indicate that 103 of them presented new testing techniques, 11 were about overview of SOA testing issues and new challenges, 4 were literature surveys, 7 discussed new SOA related measures, while only 2 references discussed *process* dimension of SOA testing. This lack of research on the process aspect motivated us to focus our attention on analyzing this critical component of a SOA development process.

In continuation of this research, establishment of a specialized and effective testing process covering these SOA testing methods, tools, and other concerns for SOA-based systems, therefore, has become a strong necessity. The assessment as to the effectiveness and efficiency of this testing process is equally important. In the next sections we show how we can evaluate quality of such kind of test processes.

3 Generic Test Process Evaluation

Considering the deficiencies of existing test process assessment and improvement models, a generic meta-measurement framework for test processes has been recently proposed [7]. This framework intends to cope with implicity and partiality of current approaches and serves as a complementary contribution in this research direction. It is based around the idea of product quality evaluation as specified in the ISO 9126 standard. The six elements of this framework, *target,*

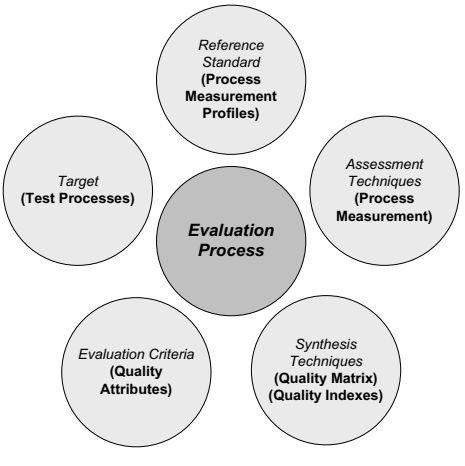


Fig. 2. Evaluation Framework

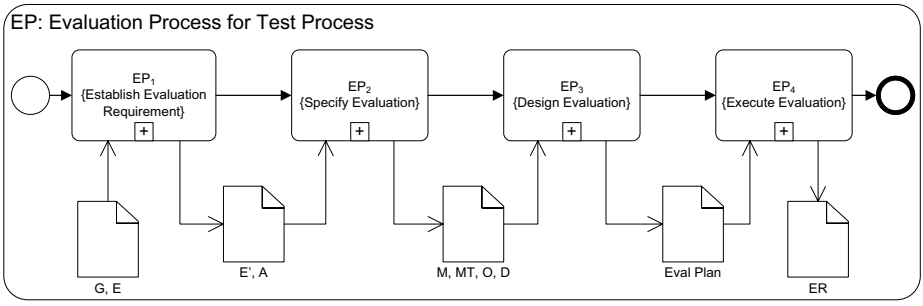


Fig. 3. Evaluation Process

evaluation criteria, reference standard, assessment techniques, synthesis techniques, and *evaluation process* have been shown in figure 2.

Skipping the discussion on any other component of this framework, here we review only its *evaluation process* element which is relevant in understanding the needed SOA testing perspectives for implementing this proposed approach. The evaluation process consists of four main steps as seen in figure 3 and shortly described below;

- **Establish evaluation requirements**
  This step involves initial planning for the evaluation. It concerns with determining the evaluation requirements based on business and strategic goals, test specifications, and stakeholder preferences. The requirements have to be organized in the form of a quality model consisting of appropriate quality attributes and sub-attributes. As a starting point, the framework defines some typical set of test process quality attributes and sub-attributes.
- **Specify evaluation**
  This step mentions defining evaluation scope. Entities of the test processes to be measured are to be identified. This entity list should be connected with required quality attributes in a quality matrix which captures relevance of each attribute/sub-attribute against an entity using weights. In this step metrics relating to each of the sub-attributes should be defined. Again, the framework provides an example set of test process metrics to be complemented by scenario specific test metrics.
- **Design evaluation**
  Procedures for the evaluation activities with required resource utilization are detailed in this step. This phase concerns with *what, when* and *how* dimensions of the evaluation process.
- **Execute evaluation**
  It consists of taking process measurements and using them to determine quantitative process quality scores using a predefined quality indexing scheme. The framework also specifies presentation of quality evaluation reports so as to derive improvement suggestions and locate weak process areas.

4 Challenges to Framework Customization

Since the framework described above is quite an abstract representation of test process evaluations, it must be implemented in an appropriate way for specific implementations of test processes. One challenge to the framework itself is its formulation as a concrete evaluation process to be embedded into an existing testing/development process. A suitable measurement tool will certainly be needed to support the measurement capturing and execution & analysis of measurement data. Considering the impact of SOA on development lifecycle activities [10] and hence on testing activities, below we discuss some situations which need special considerations while using the evaluation framework for SOA test processes.

- **Establishment of SOA Test Processes**
  The first step of the proposed framework requires identification of test process artifacts which are to be evaluated. For this purpose, a clearly defined test process should exist detailing the procedures, products, and resources involved. SOA testing may be performed for a certain objective (functional, non-functional testing), at some level (unit, integration, system), or as regression testing [6] [8]. Each of these test phases can be considered as a test process itself subject to evaluation. A helpful starting point for defining a SOA test process is a description of testing strategies for SOA based systems given by Linthicum and Murphy [11]. However, they focus mainly on setting testing goals and presenting an overall structure of the testing process excluding the list of possible test procedures and activities etc.
- **Setting Evaluation Goals**
  The next step in the evaluation framework is determination of measurement goals, i.e. what should the test process achieve. Although a pre-provided set of generic evaluation goals is provided in the framework, it suggests a customization of these goals to specific situations. Service-level agreements and typical quality attributes of SOA systems [12] such as interoperability, performance, reliability, availability etc should provide inspirations for setting test process goals.
- **SOA Test Metrics**
  Another crucial element of the proposed evaluation scheme is the test process measures. A collection of test process measures is already presented by the framework as an example set of candidate process measures. Although the list is quite exhaustive, yet many additional SOA related measures will be needed corresponding to the chosen evaluation criteria. One issue within this context is that research on SOA metrics is still in its infancy with few proposed metrics sets such as [13] [14] [15] whose validation is still pending.

5 Conclusion and Future Work

We have shortly described an existing meta level measurement framework for evaluating software test processes. The framework is designed to be a generic

approach for concrete implementation in particular environments. Considering the specialized needs and constraints of SOA testing, we have discussed various issues in applying this framework for evaluating SOA testing processes. Based on the mentioned framework, we are working on deriving a meta-model of evaluation process using SPEM [1] process modeling language to be embedded into any existing SOA test/development process. Development of a compliant measurement and evaluation support tool is also planned for validation of the proposed approach.

References

1. Erl, T.: SOA Principles of Service Design. Prentice Hall PTR, Upper Saddle River (2007)
2. Stojanovic, Z., Dahanayake, A.: Service-oriented Software System Engineering Challenges and Practices. Idea Group Inc., Hershey (2005)
3. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. Computer 40(11), 38–45 (2007)
4. Kontogiannis, K., Lewis, G.A., Smith, D.B.: A research agenda for service-oriented architecture. In: SDSOA 2008: Proceedings of the 2nd international workshop on Systems development in SOA environments, pp. 1–6. ACM, New York (2008)
5. Dustdar, S., Haslinger, S.: Testing of service-oriented architectures - a practical approach. In: Weske, M., Liggesmeyer, P. (eds.) NODe 2004. LNCS, vol. 3263, pp. 97–109. Springer, Heidelberg (2004)
6. Ribarov, L., Manova, I., Ilieva, S.: Testing in a service-oriented world. In: InfoTech 2007: Proceedings of the International Conference on Information Technologies (2007)
7. Farooq, A., Schmietendorf, A., Dumke, R.R.: A quantitative evaluation framework for software test process. In: CONQUEST 2008: Proceedings of the International Conference on Quality Engineering in Software Technology, Aachen, Germany. Shaker Verlag GmbH (2008)
8. Canfora, G., Penta, M.D.: Testing services and service-centric systems: Challenges and opportunities. IT Professional 8(2), 10–17 (2006)
9. Parveen, T., Tilley, S.: A research agenda for testing SOA-based systems. In: Proceeding of 2008 2nd Annual IEEE Systems Conference, pp. 1–6. IEEE Computer Society, Los Alamitos (2008)
10. Lewis, G.A., Morris, E., Simanta, S., Wrage, L.: Effects of service-oriented architecture on software development lifecycle activities. Software Process: Improvement and Practice 13(2), 135–144 (2008)
11. Linthicum, D.S., Murphy, J.: Key Strategies for SOA Testing. Mindreef Inc., Hollis (2007)
12. O'Brien, L., Merson, P., Bass, L.: Quality attributes for service-oriented architectures. In: SDSOA 2007: Proceedings of the International Workshop on Systems Development in SOA Environments, Washington, DC, USA, p. 3. IEEE Computer Society, Los Alamitos (2007)
13. Rud, D., Schmietendorf, A., Dumke, R.R.: Product metrics for service-oriented infrastructures. In: IWSM/MetriKon 2006: Proceedings of the International Workshop on Software Measurement and DASMA Software Metrik Kongress, Aachen, Germany, pp. 161–174. Shaker Verlag GmbH (2006)

[1] Software Process Engineering Metamodel Specification by OMG.

14. Rud, D., Dumke, A.S.R.R.: Resource metrics for service-oriented infrastructures. In: SEMSOA 2007: Workshop on Software Engineering Methods for Service Oriented Architecture, pp. 90–98 (2007)
15. Rud, D., Mencke, S., Schmietendorf, A., Dumke, R.R.: Granularitätsmetriken für serviceorientierte architekturen. In: MetriKon 2007: Proceedings of the DASMA Software Metrik Kongress, Kaiserslautern, Germany, pp. 297–308. Shaker Verlag GmbH (2007)

Criteria to Compare Cloud Computing with Current Database Technology

Jean-Daniel Cryans, Alain April, and Alain Abran

École de Technologie Supérieure, 1100 rue Notre-Dame Ouest
Montréal, Québec, Canada
jean-daniel.cryans.1@ens.etsmtl.ca
{alain.april,alain.abran}@etsmtl.ca

Abstract. After Google published their first paper on their software infrastructure in October 2003, the open-source community quickly began working on similar free solutions. Yahoo! is now able to process terabytes of data daily using Hadoop, which is a scalable distributed file system and an open-source implementation of Google's MapReduce. HBase, a distributed database that uses Hadoop, enables the reliable storage of structured data, just like Google's Bigtable which powers applications like Google Maps and Google Analytics, to name only two. Many companies are tempted to use these technologies, but it is currently difficult to compare today's systems with systems built on top of HBase. This paper presents this new technology and, a list of proposed comparison elements to existing database technology as well as proposed comparison assessment criteria.

Keywords: Cloud Computing, Bigtable, HBase, Hadoop.

1 Introduction

After Google published their first paper on their proprietary software infrastructure in October 2003 [1], the open-source community quickly began working on similar open-source solutions. Today, Yahoo! and many other companies are able to process terabytes of data daily using the open-source solution called the Hadoop [2] framework. HBase [3], a sub project of Hadoop, is a distributed database built on the same specifications as Google's Bigtable [4] which powers applications like Google Maps and Google Analytics, to name only two. Reasons to try to learn and understand this technology include eliminating licensing costs, achieving scalability and better control over the performance characteristics of the applications. When it takes a few months for a typical organization to choose, order, install, and set up a few servers, it is already too late when your Web site is growing by 30 million pages per day. This situation was faced by YouTube during 2006 [5], and they turned to the Bigtable technology to solve their issues when they were bought by Google.

Installing and using HBase is not something we learn to do in school; in other words, these tasks are not, at first glance, intuitive. At the bottom of the infrastructure illustrated in Fig. 1, there is a distributed file system designed to scale to thousands of

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 114–126, 2008.

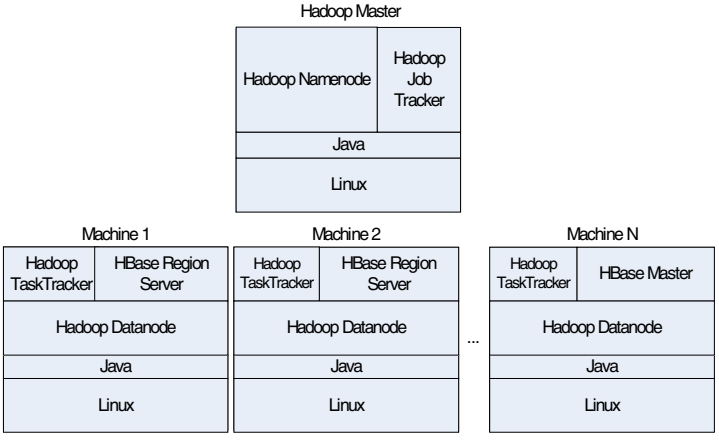


Fig. 1. HBase infrastructure (master and region servers)

machines. This is the Google File System [6] at Google and the Hadoop Distributed File System in the open-source community. They are both tolerant to machine failure and replicate data which can be counted in petabytes. To query such huge datasets, Google invented a new programming model called MapReduce [7], the idea behind it being that processing data in a distributed environment always involves the same two basic steps: 1) mapping the data needed; and 2) aggregating those data. Now, trying to move such datasets on a network would easily saturate its capacity. So, unlike supercomputing, processing is performed where the data are located -- that is, on each node -- and must be managed by a master scheduler and a number of scheduler slaves. However, a file system alone is inefficient at handling structured data, which is why Google created Bigtable, a distributed database that leverages the distributed file system. Bigtable, and its open-source equivalent, HBase, offer a simple and dynamic data model that is not relational.

Using this infrastructure implies better availability, as well as scalability and new tools to process huge amounts of data, making it is useful for large corporations, but also for startups, whose aim is to develop a large amount of traffic. Those who wish to migrate from a typical open-source relational database management system (RDBMS) to HBase currently have no tools or methodologies with which to compare the available systems. Our paper is aimed at helping these companies by presenting a list of comparison elements and demonstrating how they translate into assessment criteria. This research was carried out in parallel with a project in which our lab was working on migrating a system based on PostgreSQL for a Canadian telephone company. In section 2, we describe the Hadoop distributed file system, MapReduce, and HBase function on the conceptual level, and provide examples. In section 3, the comparison elements are defined. In section 4, we describe how the technique is used to transform comparison elements into assessment criteria. Finally, we conclude the paper with a summary and a description of future work.

2 An overview of HBase and Its Underlying Infrastructure

2.1 Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) is a distributed file system for large data-intensive applications, and is designed to run on commodity hardware. It is similar in many ways to existing distributed file systems, most notably the Google File System (GFS), and shares the same goals, such as performance, reliability, and availability. The HDFS initially served as the infrastructure for the Apache Nutch [10] Web search engine project, and is now part of the Apache Hadoop Core project. Its core developers come from Yahoo!'s Grid Team, which currently has the largest HDFS cluster (a group of computers working on a common computation) in production; it runs on more than 10,000 processors and stores over 5 petabytes of information [11].

Assumptions and Goals

- The clusters are built from commodity machines, the components of which often fail. The file system must detect faults from which it should automatically and quickly recover.
- The applications that need to access the data stored in the HDFS need streaming access. They are not general-purpose applications that connect to POSIX-compliant file systems. They rely on the HDFS for batch processing and not for interactive use by users.
- The applications that run on the HDFS have medium to large datasets. The typical file stored is gigabytes to terabytes in size. The HDFS supports small files, but is not optimized for them.
- Moving computations is cheaper than moving data; thus, running computations is much more efficient if executed near the data on which the computation operates. This is especially true with gigabytes of data. It prevents saturation of the network and increases the aggregated throughput of the system.
- The applications that run on the HDFS come from different platforms, so it is designed to be portable from one platform to another. This facilitates widespread adoption of the HDFS as the distributed file system of choice for a large set of applications.

Architecture

The HDFS is based on master/slave replication. As illustrated in Fig. 2, the master server is called a Namenode, and it both manages the file system namespace and regulates client access. The other nodes are called Datanodes, and they manage storage attached to the nodes on which they run. That storage is composed of files which are split into blocks, and it is the duty of the Namenodes to determine the mapping of the blocks to the Datanodes. Although the HDFS is not POSIX-compliant, its file system still supports the usual operations of creating, deleting, opening closing, reading, and writing files.

The typical HDFS cluster is built on commodity machines which run a GNU/Linux operating system. But since the HDFS is written in the Java language, any machine supporting Java can run a Namenode or a Datanode. It is recommended that the

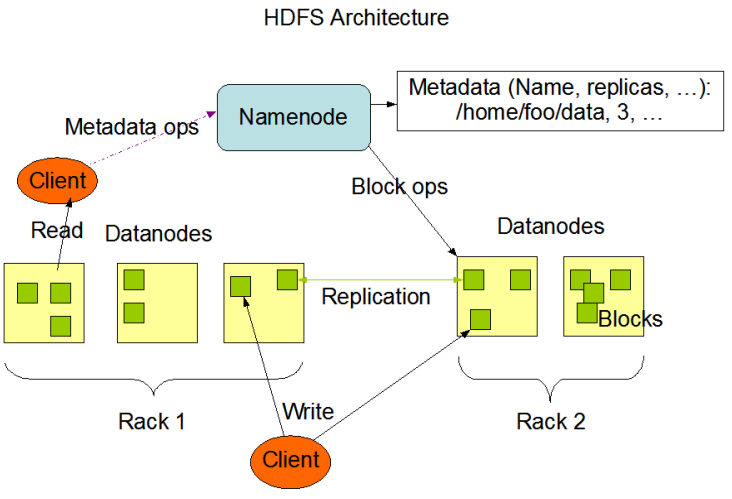


Fig. 2. The HDFS architecture. The file blocks are distributed among the Datanodes and monitored by the Namenode. The clients read and write directly from the Datanodes [12].

machine that hosts the Namenode be dedicated to that Namenode, and does not host a Datanode for main memory considerations. The other machines host only one Datanode, as running multiple instances greatly reduces the reliability of the system.

Moreover, having only one Namenode greatly simplifies the architecture of the system. In order to ensure that a bottleneck is not created, clients never read or write files through the Namenode. Instead, they ask the Namenode which Datanodes are hosting the blocks associated with the files they need.

2.2 MapReduce

The MapReduce programming model was invented by Google for processing large datasets across hundreds or thousands of machines. The software framework hides the details of computation parallelization, data distribution, and failure handling, and lets users specify a map function which transforms a set of key/value pairs into a set of intermediate key/value pairs and a reduce function that merges all intermediate values associated with the same intermediate key. The Apache Hadoop Core project contains a Java implementation of the MapReduce specification.

Programming Model

The Map function is written by the user and takes as input a key/value pair and outputs an intermediate key/value pair set. The Reduce function is also written by the user and accepts an intermediate key and a set of values for that key. The values are merged in such a way that only an output value of zero or one is produced per reducer.

The Map function will output each word in a document with an associated count of occurrences which, in this function, will always be '1'. The Reduce function will then, for each word outputted, sum its occurrences and output that figure.

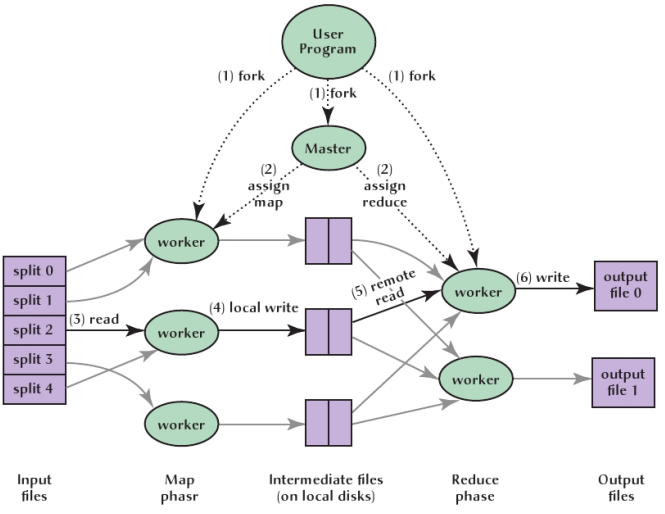


Fig. 3. The MapReduce execution flow in Google's implementation [7]

Implementation

The Hadoop MapReduce implementation [22] greatly resembles that of Google, since both are designed to handle petabytes of data daily. The typical cluster consists of the same types of machine used for the HDFS, and uses it as its distributed file system. The jobs, a set of Map and Reduce functions, are submitted to a scheduling system to the available machines.

In summary, the execution flow proceeds in the following sequence (numbering refers to the numbers in Fig. 3):

1. The Map functions are started for each input file split on its hosting machines.
2. A special node, the JobTracker (or Google's Master), assigns a Map or a Reduce task to each idle TaskTracker (or Google's Worker).
3. The TaskTracker then reads the block it was assigned, and parses each key/value pair out of the input data and passes it on to the Map function.
4. The Map outputs are then sorted and partitioned per Reduce function. A Combine function is an optional which can be used to aggregate the outputs of each Map of each machine, thereby limiting data transfer over the network.
5. When a TaskTracker responsible for a Reduce is notified that mapped data are available, it reads its content via HTTP. As the data are retrieved, they are sorted by the intermediate key.
6. Once the intermediate keys are sorted, the Reduce TaskTracker iterates over each unique key and passes it along with its set of intermediate values to the Reduce function. The output is appended to a final output file for this Reduce partition.

Usage

Google uses MapReduce in a variety of internal applications, and states that, in September 2007, 11,081 machines were used in MapReduce jobs [13].

Yahoo! uses Hadoop in their Web search, and the output of their MapReduce tasks is over 300 terabytes, compressed [11].

Rackspace uses Hadoop to parse its logs generated from the machines of their datacenters to collect statistics (such as spam counts) from their mail system [14].

Facebook uses Hadoop in a 320-machine cluster to store internal log and dimension data sources, as well as for reporting, analytics, and machine learning [15].

2.3 HBase

HBase is an open-source, distributed, column-oriented, multi-dimensional, high-availability, high-performance storage technology written in Java and inspired by Google's Bigtable. Just as Bigtable leverages the distributed data storage provided by the Google File System (GFS), HBase is aimed at providing Bigtable-like capabilities on top of the HDFS. HBase has been a part of the Apache Hadoop project since February 2008, and is used in production environments.

Motivations

Scaling out a typical relational database often begins with replication. For example, YouTube [5] first used MySQL master-slave replication, but eventually arrived at a point where the writes used all the capacity of the slaves. Like many organizations, they tried partitioning their tables into shards, so that sets of machines hosting the various databases were optimized for their tasks. Flicker [19] did the same thing with their shards, and even duplicated some data since the comments table was linked to their user shard and to their images shard. Soon, after going through many difficult steps to scale the architecture, they found that the first relational solution became denormalized and harder to maintain. Tools like Pyshards [17], aimed at easing shard management, were developed, but they do not obviate the need for partitioning a shard if it becomes too big.

Goals

To quote the authors of HBase, "The HBase project is for those whose yearly Oracle license fees approach the GNP of a small country or whose MySQL install is starting to buckle because tables have a few BLOB columns and the row count is heading north of a couple of million rows." [18] HBase is designed to leverage the HDFS to reliably store terabytes of sparse structured data. It should be as usable in real-time applications, where low latency is the priority, as in batch jobs, where higher throughput is important. HBase should not provide a relational model, but instead consist of a simple data model which supports dynamic modifications of data layout.

Data Model

HBase, like Bigtable, is a sparse, distributed, and persistent multi-dimensional sorted map which is indexed by a row key, a column key, and a timestamp, and the value is an uninterpreted array of bytes, as illustrated in Fig. 4. The column keys reside in column families, which are the same for each row and have the syntax "family:qualifier".

$$(row{:}string, column{:}string, time{:}int64) \rightarrow string$$

Fig. 4. Representation of HBase indexation of the values [4]

This map is persistent because it is stored in HDFS, which also makes it distributed. It is sorted because the row keys, not their values, are maintained in a lexicographical order in the cluster. It is multi-dimensional because the value of each row's key/column key pair has a timestamp; it is sparse because the column keys in column families are not necessarily the same across rows; and it is distributed because its components reside on many machines.

The database schema in HBase and Bigtable is very different from the relational schema. First, there are no joins between tables. For example, a blog would have three tables: blogs have many comments and many users. A solution for the blog/comment relation would be to have column families for each comment property, and the column key would be the comment ID. If the comments have properties that do not need to be accessed with their blog entries (such as the Web browser that was used), a second table can be created containing all the information. Since the system runs on commodity hardware, the disks are cheap and space is abundant.

The schema also has only one index, the row key. Unlike the RDBMS, the developers cannot use the equivalent of the WHERE clause. By design, all access methods in HBase are either gets or scans over the row key, so that a query cannot slow the whole system down. Since HBase sits on the Hadoop framework, MapReduce can be used instead to generate index tables.

HBase column families can be configured to support different types of access patterns. A maximum number of cell timestamps can be specified, as well as a time to live (TTL). By default, HBase returns the latest cell version available, keeps only one version, and has no TTL. A family can also be set to stay in-memory, yielding better performance at the expense of consuming the main memory. Also, since the data in a family are typically of the same nature, a compression level can be configured. For example, compressing a family that stores many versions of big strings which are not read often will leave more space on the disk for other, more important information.

Architecture

An HBase cluster is composed of individual Master and Region Servers which can be counted in the hundreds. The Master's job is to monitor the load and to coordinate the Region Servers. Each Region Server hosts a number of Regions, which it stores in the HDFS. A Region is composed of sorted rows from a table, so that all table rows are contained in a set of Regions. HBase relies on Zookeeper [16], which, like Bigtable, relies on Chuggy[8], to ensure that there is always one Master at any time, to store the bootstrap location of HBase data and to discover the Region servers.

The way the architecture works is based on the fact that the rows are ordered by row key. A three-level hierarchy, similar to a B+ tree, is used to store the Region locations. The first level is a file stored in Zookeeper that contains the location of the ROOT Region that the Master reads during its startup. That Region then gives the location of all the other META table's Regions, which in turn give the location of all the user Regions. So, when an application requests a particular row, it first reads the META table to figure out in which Region the row is located and then directly contacts the server hosting the Region to retrieve the row. There are many caching levels, but these are not discussed in this paper. The writes are processed in a similar fashion. Instead of figuring out which Region hosts the row, it will figure out which Region *should* host it, once it is sorted. If a Region becomes too big according to a configured threshold, it will be split into two.

Accessibility
HBase is accessible from its Java Application Programming Interface (API) which offers Create, Read, Update, and Delete (CRUD) operations as well as cluster management tools. The following program code demonstrates its use. The first operation shows how to open an access to an HBase table and fetch a value using a row and a column key. The second operation shows how to open a transaction on a row, insert a new value under a column key, and delete another value.

```
(1)HTable table = new HTable("myTable");
   byte[] valueBytes = table.get("myRow",
      new Text("myColumnFamily:columnQualifier1"));
   String valueStr = new String(valueBytes);

(2)long lockId = table.startUpdate("myRow");
   table.put(lockId, "myColumnFamily:columnQualifier1",
      "columnQualifier1 value!");
   table.delete(lockId,
"myColumnFamily:cellIWantDeleted");
   table.commit(lockId);
```

HBase can also be accessed via a REST gateway, which supports the statements GET and PUT which, when executed, return data in XML format. Another way to communicate with HBase is via a Thrift [20] gateway. Thrift is a framework enabling cross-language Remote Procedure Calls (RPCs) and was developed by Facebook. This means that most of the APIs can be accessed by the following programming languages: C++, Ruby, Python, and PHP.

3 Comparison Elements

We investigate a case study where an existing system, using RDBMS technology, has been migrated to the HBASE technology. In our lab, both systems are operating in a controlled environment. Our goal here is to compare the two systems and draw some conclusions on the usefulness of this new technology. In comparing the systems, many factors need to be taken into account. Initially, it appeared that the comparison would be difficult, as some elements do not vary, while others are very different. Below is a discussion on the first set of comparative elements we would like to assess:

- **Software architecture:** The software architecture could remain the same for each implementation. Typically, modern applications will have a presentation layer which communicates with a business logic layer, which in turn communicates with a database layer. In migrating to HBASE, only the database layer should need adaptation.
- **Hardware:** The hardware in the two implementations was definitively different. Using an RDBMS, the typical machines for a website with a sizeable traffic volume will be composed of as many processors and as much memory as can be fitted onto a motherboard. These components will also be server-class, since the consequences of a machine failure on availability are dire. This equipment is also expensive. The disks are organized in the RAID (redundant array of independent disk) technology that maximizes writing and

reading speed, along with replication. That equipment is installed in a master-slave scheme recommended by the manufacturer of the RDBMS. If the cost of replicating the write operations becomes too great, the database will be partitioned into shards.

In contrast, the equipment typically used for HBase is PC-class, so they are not very expensive. The recommended components are a dual or a quad processor with as many gigabytes of memory as possible and two hard drives of at least 250 gigabytes. All these machines will work together in a cluster which can consist of from 1 to 500 machines.

- **Operating system:** The majority of RDBMSs can be deployed on many operating systems, but HBase is currently only supported on the Linux distributions. This has to be taken into account if the RDBMS is not hosted on Linux.
- **Data structure:** The two data structures are very different. It is assumed that the reader is familiar with the relational schema, so it will not be described here. The Bigtable/HBase schema is described in section 2. The main differences are that all the relational primitives have been relaxed to allow a more dynamic table schema, and that there is no full scan available in HBase. This may seem restrictive from the point of view of a developer of small systems, but the data-intensive expert will recognize that this is also not doable in a relational system because the joins cannot all be performed in memory (unless the database was denormalized, which requires much more disk space).
- **Data manipulation:** Data manipulation is very rich and mature for online operations in an RDBMS and very restricted in HBase. It is assumed that the reader is familiar with SQL, so it will not be described here. The data manipulation for HBase is described in section 2. The biggest advantage of using HBase is that it is enabled to work with MapReduce as a source or as a sink, or both. Instead of performing the equivalent of a JOIN at runtime, it is more convenient in HBase to build an index table. For example, Bigtable's paper describes the way to generate the many reports in Google Analytics, which is to run a nightly MapReduce job which takes a table of clicks from each website as a source and an aggregate table as a sink.
- **Means to scale:** Scaling an RDBMS first means upgrading each machine, and, since these machines are expensive, the whole process is even more so. This method has a physical limit, which is the current state of the technology, and, unless there are redundant databases with custom load balancing, a downtime has to be scheduled. Scaling can also be performed with replication, so that each slave can handle the read operations while the master receives the writes. It is a method which works until the replication of each write on each server generates a bottleneck. This problem is solved by sharding the database, but an additional layer of software is needed to handle the shards. In any case, all these operations are very expensive.

  Scaling HBase is simply done by adding more machines. For each new machine, this is a matter of installing the operating system and the software, and then starting the Hadoop and HBase processes. The Datanode/Region servers will contact their respective master node, and, in return, will begin redistributing the load. The software that runs in the cloud will not be impacted.

- **Where hardware reliability is handled:** Hardware reliability is a big issue with any RDBMS, since these systems are designed to work on only one machine initially. This is why expensive solutions have to be designed to ensure that a machine failure will not take the whole system down. Having server-class components provides lower mean time between failure (MBTF), but it does not shield the system from a defaulting motherboard, which is why failover mechanisms are designed. These mechanisms can, for example, hold off the writes in case the master machine in a replication scheme dies to give system administrators time to modify the DNS to point to a new master. If a slave dies, a big part of the throughput goes down because of the small number of machines.

  In HBase, reliability is handled at the file system and database levels. If any slave machine dies, only a small part of it goes down, and this is considered "normal", given the number of machines. For example, bigger clusters like the ones they have at Yahoo! are able to sustain losing a whole rack.
- **How many systems are using the system:** A very important feature of cloud computing is that a single cloud can hold many systems, even if they have different quality requirements. It is economically wise to aggregate systems in a single cluster, because then every system will benefit from a newly added machine. In contrast, with the RDBMS, there are different databases for different applications, since scaling is difficult and expensive. When taking measures, one has to make sure to clearly define what in the cloud can be "given" to the software that is being compared to an RDBMS implementation.

This research was carried out in parallel with a project in which an existing system based on PostgreSQL had to be migrated to a system based on HBase. While choosing the hardware, the setup followed was the one that Google uses, as described in the Bigtable paper [4], and it was very different from that already in use. For example, the typical machine for HBase had dual processors, while those used in the source system were dual quad processors. To minimize the differences, the same operating system was kept, the system's architecture was not changed, and the cluster

Table 1. Comparison of elements proposed in the case study

Comparison element	PostgreSQL implementation	HBase implementation
Software Architecture	Three-tier	Three-tier
Hardware	Few, expensive	Scores, commodity
Operating System	Cent OS	Cent OS
Data structure	Relational tables	Bigtable-like structures
Data manipulation	ORM	HBase client API, MapReduce
Means to scale	Expensive	Inexpensive
Where hardware reliability is handled	Custom solution	HBase and Hadoop
How many systems are using it	One	One

was only used by the target system. By using Hadoop and HBase, the scalability and back end reliability requirements were easily met, so there was no need to keep the custom solution to handle them. Since the source system was in Java, the HBase client API was used directly. Table 1 summarizes the differences identified in each element during the research.

4 Transforming Comparison Elements into Comparison Criteria

To transform the comparison elements into criteria for a comparison assessment, a two-step technique was used. The process is presented below:

1. Identify impact: The impact that a comparison element has when it is not the same in the two implementations will lead to different measurement results.
2. Identify the direct effects on quality: Each comparison element has different effects on internal and external quality, as defined in the ISO/IEC 9126 models of software product quality [23].

Table 2 illustrates an application of this technique to derive assessment criteria for comparison.

Table 2. Assessment criteria mapping

Comparison element	Impact	Related ISO 9126 quality sub-characteristics
Software Architecture	Changing the architecture implies completely different quality attributes that need to be evaluated.	All criteria
Hardware	Commodity hardware is less reliable and, on its own, performs poorly.	Fault tolerance Time behavior Scalability
Operating System	If the source operating system is not Linux, its efficiency and maturity are different.	Fault tolerance Time behavior
	Restricting to Linux makes the system less adaptable.	Resource behavior Adaptability
Data structure	The target data structure is easy to change and provides constant performance.	Time behavior Changeability
	This data structure is not relational and not taught in classes.	Replaceability Adaptability
Data manipulation	The target data manipulation provides limited functionalities for online processing, but is excellent for offline processing.	Time behavior Analyzability Changeability
	MapReduce is unknown to most developers.	Replaceability
Means to scale	Scalability is built into Hbase, provides for easier reactions to new specifications, and does not require the system to be shut down.	Scalability Stability Adaptability
Where hardware reliability is handled	Reliability is also built in, so there is no need for a custom layer to provide it. The system is simplified.	Reliability Analyzability Changeability
How many systems are using it	The target system may perform badly if other systems are heavy users of the resources.	Time behavior

In Table 2, each row constitutes an assessment criterion which helps in comparing the quality of two systems. As described, HBase use may negatively affect the maintainability of the system, since its current and future developers may not be experts in Hadoop and HBase. At the same time, it provides a system that does not need to handle database failures. Moreover, it enables faster processing of large datasets and can scale according to changing requirements.

5 Future Work and Summary

While new technologies have been developed in the last few years to address the scalability and reliability problems inherent in data-intensive systems, little has been done to validate the quality of the new systems relative to the older ones. Based on our work in migrating a system that used PostgreSQL to one that used HBase, a list of comparison elements was identified and translated into assessment criteria. We do not claim that the list is exhaustive, and more research should be done to verify and validate the criteria it contains.

Future work is aimed at developing a process to measure the scalability and performance criteria. The only measures published for Bigtable-like databases to date are presented in Google's original paper [4]. Google used N machines and took measures as N varied while N clients were generating a load. The following measures were reported: random reads, random reads in memory, random writes, sequential reads, sequential writes, and scans for 1, 20, 250, and 500 machines. Results were impacted by the fact that the machines were also used by other processes during the experiments.

References

1. Carr, D.: How Google Works, http://www.baselinemag.com/c/a/Projects-Networks-and-Storage/How-Google-Works-%5B1%5D/
2. HADOOP.COM. Product page, http://www.hadoop.com
3. HBASE.ORG. Product page, http://www.hbase.org
4. Chang, F., Dean, J., Ghemawat, S., et al.: Bigtable: A Distributed Storage System for Structured Data. In: 7th Symposium on Operating Systems Design and Implementation (OSDI 2006), Seattle, WA, USA, November 6-8, pp. 205–218 (2006)
5. Cordes, K.: YouTube scalability Talk (July 14, 2007), http://kylecordes.com/2007/07/12/youtube-scalability/
6. Ghemawat, S., Gobioff, H., Leung, S.-T.: The Google file system. In: Proc. of the 19th ACM SOSP, December 2003, pp. 29–43 (2003)
7. Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. In: Proc. of the 6th OSDI, December 2004, pp. 137–150 (2004)
8. Burrows, M.: The Chubby lock service for loosely coupled distributed systems. In: Proc. of the 7th OSDI (November 2006)
9. YAHOO.COM. Product page, http://research.yahoo.com/node/1849
10. APACHE.ORG. Product page, http://lucene.apache.org/nutch
11. Baldeschwieler, E.: Yahoo! Launches world's biggest Hadoop production application (February 19, 2008), http://developer.yahoo.com/blogs/hadoop/2008/02/yahoo-worlds-largest-production-hadoop.html

12. Borthakur, D.: The Hadoop Distributed File System: Architecture and Design (May 21, 2008), http://hadoop.apache.org/core/docs/r0.17.0/hdfs_design.html
13. Dean, J., Ghemawat, S.: MapReduce: Simplified data processing on large clusters. Communications of the ACM 51(1) (January 2008)
14. Hood, S.: MapReduce at Rackspace (January 23, 2008), http://blog.racklabs.com/?p=66
15. List of companies using Hadoop (June 6, 2008), http://wiki.apache.org/hadoop/PoweredBy
16. Reed, B.: Zookeeper (March 25, 2008), http://research.yahoo.com/node/2120
17. GOOGLE.COM. Product page, http://code.google.com/p/pyshards/wiki/Pyshards
18. Delap, S.: HBase Leads Discuss Hadoop, BigTable and Distributed Databases (April 28, 2008), http://www.infoq.com/news/2008/04/hbase-interview
19. Hoff, T.: How to learn to stop worrying and use lots of disk space to scale (May 21, 2008), http://highscalability.com/how-i-learned-stop-worrying-and-love-using-lot-disk-space-scale
20. FACEBOOK.COM. Product page, http://developers.facebook.com/thrift/
21. HADOOP.CA. Product page, http://www.hadoop.ca
22. Unknown author, Hadoop MapReduce Tutorial (May 21, 2008), http://hadoop.apache.org/core/docs/r0.17.0/mapred_tutorial.html
23. ISO/IEC 9126:1999. Software Engineering – Product quality. Int. Org. for Standardization, ISO 9126 (1999)

Comparison of Process Quality Characteristics Based on Change Request Data

Holger Schackmann and Horst Lichter

RWTH Aachen University, Research Group Software Construction
Ahornstr. 55, 52074 Aachen, Germany
{Schackmann,Lichter}@swc.rwth-aachen.de

Abstract. The evaluation of metrics on data available in change request management (CRM) systems offers valuable information for the assessment of process quality characteristics. The definition of appropriate metrics that consider the underlying change request workflow and address the information needs of an organization is an intricate task.

Furthermore CRM systems usually provide only a number of predefined metrics with limited adaptability. The tool BugzillaMetrics offers a more flexible approach that simplifies defining and adjusting new metrics. However a systematic approach for deriving an appropriate metric in a target-oriented way is needed. This paper describes a corresponding procedure on how to develop and validate metrics on CRM data applicable for the comparison of process quality characteristics.

Keywords: Process Metrics, Change Request Management, Metric Specification, Software Measurement Design, Measurement Tool.

1 Introduction

The management of a large software project portfolio raises several managerial challenges, like balancing resource allocation between different projects, and aligning development processes to the standards of the organization. Hence the project statuses and process quality characteristics, like planning precision or problem resolution speed, must be monitored continuously in order to identify development process weaknesses, and assess process improvements. Collecting the required data by regularly project status reporting can be expensive and intrusive, and furthermore ignores the past history of the process [1]. This motivates mining data from routinely collected repositories like change request management (**CRM**) systems.

The usage of this data for evaluating process quality characteristics imposes certain difficulties. Appropriate metrics will depend on the designated process and the improvement goals, as well as on the data available. It must be validated that the metrics are proper numerical characterizations of the qualities of interest, and that the measurements can be compared between different projects.

However existing CRM tools provide only a number of fixed metric evaluations and are limited in their adaptability [2]. Hence extraction and integration of the data

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 127–140, 2008.

typically require the development of custom scripts. Validation of the metrics most often necessitates adjusting the metrics definition and corresponding scripts, which is time-consuming and costly.

The open source tool BugzillaMetrics implements a more flexible approach for the evaluation of metrics on CRM data, based on declarative metric specifications [2]. The tool allows concentrating the main effort on the model of the metric, not on its implementation. Thus experimenting with metrics and adjusting them is faster and easier.

But, first experience with using the tool has revealed certain pitfalls in developing appropriate metric definitions [2]. This motivates the need for a structured approach for developing metrics on CRM data.

This paper presents a procedure to systematically develop metrics on CRM data used to compare process quality characteristics. This procedure includes validation steps as well as guidance for the interpretation of the metric results. First an overview of related work is given. Section 3 briefly describes the BugzillaMetrics tool.

2 Related Work

There exist numerous approaches to analyse CRM data as well as data from version control systems for several purposes (e.g. visualization of software evolution [4], or change impact analysis [5]). A survey is given by Kagdi et al [6]. Some of these approaches do also analyze specific aspects of the process. For example Sliwerski et al. present an approach to reconstruct links between the version-control system and resolved defect reports in the CRM database in order to analyse the frequency of fix-inducing changes [7]. Koponen presents a tool to analyse several aspects of maintenance processes of open source software, like typical defect-lifecycles, and origin of change analysis [8]. Gasser and Ripoche analyse CRM data of open source projects in order to extract their process models [9].

However none of these approaches is targeted at a general procedure for the assessment of process quality characteristics based on CRM data.

3 BugzillaMetrics

BugzillaMetrics is based on user defined metric specifications that abstract from the way the information is stored in the CRM database [3]. The basic building blocks for these specifications are **filters** for properties of a change request (e.g. its severity, status, or target milestone), and **events** that occur in the history of a change request (e.g. its creation, a change of the assignee, or the reopening of a resolved request). Filters and events can be combined with Boolean operators.

Each metric specification contains a **base filter** that defines which change requests are considered during the calculation (e.g. only change requests that belong to a certain product). Further on the **evaluation time period** and the **time granularity** (e.g. month or year) are defined.

Then one of several predefined **value calculators** can be applied to calculate a value for individual change requests in each time interval according to the given time

granularity. Examples of value calculators are the calculation of the length of a time interval between two specified events in the lifecycle of a change request, the calculation of the time a change request resides in a certain state, or the calculation of the number of occurrences of certain events during a time period. In the latter case an optional **weight** can be applied (e.g. a weighting by the severity of the change request, or by its estimated remaining workload). In terms of the ISO/IEC 15939 standard [10] the outcome of a value calculator can be denoted as **base measure** while a change request is the entity to be characterized by measuring.

The outcome of these value calculators can be combined with operations like sum, maximum, or mean value to calculate a result for a certain time interval. This outcome represents a **derived measure** related to the process in a certain time interval.

Thereby the tool offers a large flexibility for the specification of metrics. Furthermore the metric specification is separated from the way the required information is retrieved from the CRM database.

4 Developing Metrics on Change Request Data

This section describes a procedure on how to develop and validate metrics that target at the comparison of process quality characteristics. The approach is exemplified by developing metrics applicable to the CRM database of the Eclipse open source community.

4.1 Bidirectional Quality Models

In order to derive a metric we rely on the approach of bidirectional quality models [11]. This subsection briefly describes the related concepts (see Figure 1) and maps them to the terms of the ISO measurement information model contained in the ISO/IEC 15939 standard [10].

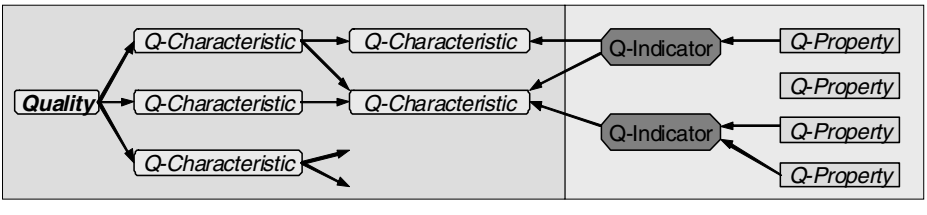


Fig. 1. Concepts of the bidirectional quality model

On the one side the **quality characteristics** reflect high-level requirements on the quality. In terms of the ISO/IEC 15939 standard these quality characteristics correspond with information needs derived from the business, organizational, regulatory, product or project objectives. An example of a quality characteristic is planning precision which can be subdivided into the quality characteristics: adherence to schedule, adherence to planned effort, and process transparency.

On the other side the **quality properties** denote objective characteristics of an entity (i.e. product, process, or system), that can be used to distinct between the considered entities. In terms of the ISO/IEC 15939 standard a quality property is an attribute of an entity that can be objectively and quantitatively distinguished by automated means. Examples for such quality properties are the total number of defects, the number of reopened change requests, or the frequency of assignee changes of a change request.

The quality properties will be used in a bottom-up fashion to form **quality indicators**. A quality indicator describes how a number of quality properties can be interpreted with respect to a quality characteristic. Hence the quality indicators bridge the gap between the technical view of quality properties and the abstract view of the quality characteristics. The notion of quality indicator complies with ISO/IEC 15939.

4.2 Identification of Quality Characteristics

The process quality characteristics of interest correspond to information needs that are in general derived from the objectives of the organization [12]. These characteristics can be refined stepwise.

For example the Eclipse community applies an agile development process based on several practices [13]. This process implicitly contains certain objectives, e.g. the planning precision of the scheduled milestones.

Related to the practice called “community involvement”, one can derive the process quality characteristic “**responsiveness to incoming defect reports**”, since the establishment of an active community requires timely reactions on observed problems. Note that this characteristic does not consider the resolution time of defects, but the duration to the first reaction on an incoming defect report. As most of the Eclipse projects are related to offering a general tools and integration platform the responsiveness on defect reports will have an impact on dependent projects based upon the Eclipse technology. In the following we will use this process quality characteristic as an example.

4.3 Identification of Quality Properties

In order to identify measurable quality properties it is necessary to analyze the way the CRM system is used, e.g. it must be examined what is the typical workflow of a change request, and which information is collected on a change request. Then quality properties need to be defined where some relation to the quality characteristics is conjectured.

Since each quality characteristic is related to some improvement goal, potential quality properties can be identified based on the Goal Question Metric approach [14]. However since the analysis is based on CRM data, it must be possible to determine a quantitative value for each quality property based on the data collected during the lifecycle of a change request. Naturally there will be some process quality characteristics where it is not possible to determine related quality properties, since not all quality characteristics can be evaluated based on the available CRM data.

For our ongoing example we need to find out which reaction can be considered as an appropriate acknowledgement for an incoming defect report. At first sight this will

be either adding an additional comment, changing the status of the defect report, or assigning the defect report to a specific assignee. These events can be specified in a metric of the BugzillaMetrics tool. The plausibility of the metric can then be validated by inspecting the results calculated for individual change requests and examining whether the history of a request conforms to the envisaged interpretation.

In our case it needs to be checked whether the metric considers all relevant reactions on defect reports. A detailed consideration shows that there are some more reactions, like changing the severity, priority, or the target milestone of the defect report, since this gives feedback about how the defect is rated by the project team. Hence the metric definition will be refined stepwise.

Until now the metric does only specify how the numbers for individual change requests are calculated. The next subsection will describe how these numbers will be aggregated in order to compare the process quality characteristics of different projects.

4.4 Definition of Quality Indicators

At first it needs to be defined how the values for individual change requests of a project can be aggregated. An appropriate metric must fulfil the following requirements:

- **Elimination of interfering factors:** It must be avoided that the aggregated value can be predominantly influenced by other factors, like size and age of the project. If other factors interfere with the original intention of the measurement the result will be difficult to interpret and to compare between projects.

  This would be the case in our example if we take the arithmetic mean of the individual first reaction values for each defect report. The result would potentially be distorted if a number of old but untreated defect reports is processed in a long-lived project. Hence the aggregation of the individual values will require using some kind of normalization or mean value that is stable against these kinds of outliers.
- **Timely relatedness to perceived problems in the process:** Each value calculated for a change request belongs to a specific time interval (e.g. month or year). It must be carefully considered that the assignment of the measurement values to time intervals stands in a temporal connection to potential causes in the process in order to prevent misleading interpretations.

  In our example this may happen if the values would be assigned to the time interval in which the first reaction occurred. Processing a number of old but untreated defect reports might then erroneously be interpreted as bad responsiveness in that time interval. Though the cause why these defect reports remained untouched dates back to the past.
- **Appropriate granularity of time intervals:** The granularity of the time intervals for which the change request values are aggregated must be similar to the release cycle of the project. Otherwise the resulting values will possibly be diverging due to the current phase in the release cycle (e.g. the endgame phase in the Eclipse development process [13]).

In order to derive an aggregated value that fulfils these requirements it is often better not to use the absolute values (in our case the time until first reaction on a change request), but to count the change requests where this value exceeds a certain threshold.

Hitting the threshold should be related to having a negative or positive impact on some quality characteristic. In our example it is reasonable to assume that defect reports with a severity equal or higher than normal that do not get any response within three days will probably retard or hamper dependent projects.

$$\text{ReactionEvents}(CR) = \{\textbf{commentAdded}(CR), \textbf{statusChange}(CR), \textbf{assigneeChange}(CR), \ \textbf{...}\} \quad (1)$$

$$\text{TimeUntilReaction}(CR) = \min\{\text{timeBetween}(\textbf{creation}(CR), e) \mid e \in \text{ReactionEvents}(CR)\} \quad (2)$$

$$\text{ThresholdHit}(CR) = \begin{cases} 0 & \text{TimeUntilReaction}(CR) \leq 3 \text{ days} \\ 1 & \text{otherwise} \end{cases} \quad (3)$$

$$\text{ThresholdHitDate}(CR) = \begin{cases} \textbf{creationDate}(CR) + 3 \text{ days} & \text{ThresholdHit}(CR) = 1 \\ \textbf{creationDate}(CR) + \text{TimeUntilReaction}(CR) & \text{ThresholdHit}(CR) = 0 \end{cases} \quad (4)$$

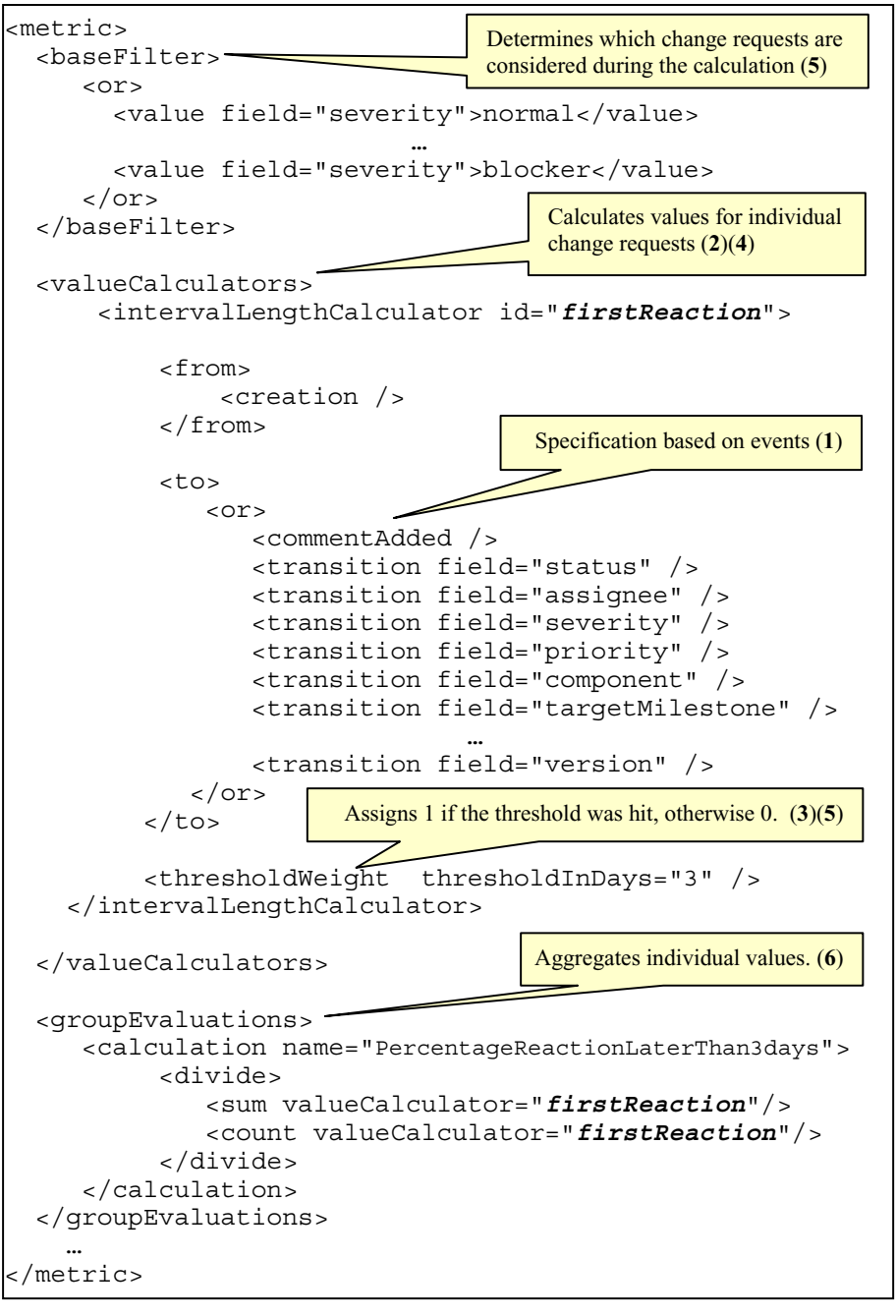
$$\text{ConsideredCRs}(\textit{timeInterval}) = \left\{ CR \;\middle|\; \begin{array}{c} \textbf{isDefectReport}(CR) \wedge \textbf{severity}(CR) \geq \textbf{\textit{normal}} \wedge \\ \text{ThresholdHitDate}(CR) \in \textit{timeInterval} \end{array} \right\} \quad (5)$$

$$\text{PercentageOfLateReactions}(\textit{timeInterval}) = \frac{\sum_{CR \in \text{ConsideredCRs}(\textit{timeInterval})} \text{ThresholdHit}(CR)}{\left|\text{ConsideredCRs}(\textit{timeInterval})\right|} \quad (6)$$

Fig. 2. Definition of PercentageOfLateReactions

Counting the change requests at the time when the threshold was hit ensures that there is a timely relation to perceived unresponsiveness. Additionally it enables to consider those defect reports in the calculation that did not yet get a response. Normalization of the results can be achieved by calculating the percentage of defect reports whose first response hits the threshold. The metric definition is sketched in Figure 2.

The calculation can again be specified with BugzillaMetrics (see Figure 3, numbers refer to the corresponding part of the metric definition). In order to use the aggregated result as quality indicator it requires defining some guidance how to interpret the results. This will be discussed in the following section.




```
<metric>
  <baseFilter>
     <or>
       <value field="severity">normal</value>
                              …
       <value field="severity">blocker</value>
     </or>
  </baseFilter>

  <valueCalculators>
      <intervalLengthCalculator id="firstReaction">

          <from>
              <creation />
          </from>

          <to>
             <or>
                <commentAdded />
                <transition field="status" />
                <transition field="assignee" />
                <transition field="severity" />
                <transition field="priority" />
                <transition field="component" />
                <transition field="targetMilestone" />
                              …
                <transition field="version" />
             </or>
          </to>

          <thresholdWeight  thresholdInDays="3" />
     </intervalLengthCalculator>

  </valueCalculators>

  <groupEvaluations>
     <calculation name="PercentageReactionLaterThan3days">
          <divide>
             <sum valueCalculator="firstReaction"/>
             <count valueCalculator="firstReaction"/>
          </divide>
     </calculation>
  </groupEvaluations>
     …
</metric>
```

Fig. 3. Metric Specification of "Percentage of Reaction later than 3 days". Numbers refer to the related formulas in Figure 2.

4.5 Interpretation Based on Empirical Data

The comparison within a peer group of projects offers a practical approach for the interpretation of the measurement values in order to decide whether a project is doing good or bad with respect to a certain quality characteristic.

The CRM system of the Eclipse project provides in our example the necessary empirical data. The resulting measurement values for a number of large projects are shown in Figure 4. Since the release dates of the major Eclipse projects are aligned in simultaneous release at the end of June, the measurement values have been calculated for the time periods between these releases.

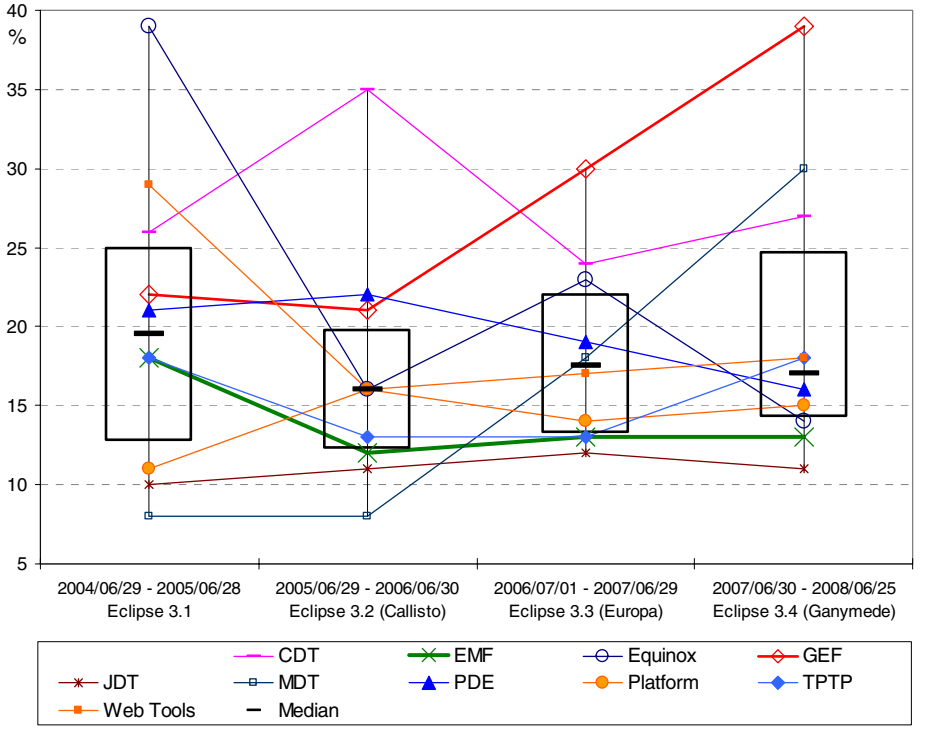


Fig. 4. Percentage of defect reports with the first reaction later than 3 days and a severity of "normal" or higher

It can be observed that the values for most of the projects tend to change only gradually between the years. This matches with the experience that discontinuous improvements of the process can rather seldom be achieved in large projects. If the values for most of the projects are volatile the underlying metric definition should be examined whether it really fulfils the requirements stated in the previous section.

A value for a project can now be interpreted in comparison to the value distribution in the time period related to the release. Naturally there can be slight differences of the interpretation dependent on the base for the comparison. The boxplot denotes the

second and third quartile of the data set. Projects within this range can be interpreted as having an around average responsiveness to incoming defect reports. So it can for example be stated that the EMF project had a good responsiveness for several release periods. The responsiveness of the GEF project is rather poor and declined in the last years. These results match with the experience gained during the development of a toolset at our research group that is called ViPER and is based on EMF and GEF [15].

4.6 Additional Example

In order to illustrate the procedure for metrics development an additional example will briefly be discussed in this section. At first we have to identify a **quality characteristic** of interest. A general objective in the development process is the efficient processing of the change requests. A related sub-goal is that the change requests should be resolved initially in an adequate way, since later rework often requires additional effort, and may be caused by insufficient coordination or misunderstandings related to the initial change request. Hence we can derive the quality characteristic **"frequency of rework"**.

In order to identify related measurable quality properties we have to analyse how rework is reflected in the available information about the lifecycle of a change request. Bugzilla has two related fields: *status* and *resolution*. If some action has been taken to resolve a change request, it is switched to the "Resolved" status. Some Eclipse projects also use the subsequent status values "Verified" and "Closed".

The resolution field indicates how the change request was resolved. Possible values are for example "Fixed" (some change to the software had been implemented), "Duplicate" (change request is already described in another existing change request), "WorksForMe" (described problem could not be reproduced), or "Not_Eclipse" (problem is related to a third-party package).

If the resolution is deemed to be incorrect the change request can be switched to the status "Reopened". A state transition to the status "Reopened" of a change request with resolution "Fixed" would indicate that a bug fix or new feature had not been implemented correctly. If the change request had a different resolution (e.g. "Duplicate" or "Not_Eclipse") this indicates that the decision to resolve the change request was based on some wrong assumptions.

Basically one can define the **quality property** "number of transitions to Reopened during the lifecycle of a change request". However we have to clarify whether we are only interested in transitions where some rework of previous changes in the software is required, or we are interested in all transitions where a change request is reexamined for some reason. The first interpretation would require taking the resolution field into account when identifying the respective state transitions. While both interpretations are reasonable, we choose the latter one here, since we are more interested in rework related to the overall change request process, instead of rework related to the quality of the implemented changes in the software.

Again, the plausibility of the previous assumptions can be validated by inspecting the lifecycle of individual change requests. By doing this we notice that the Eclipse CRM database allowed setting the resolution "Later" or "Remind" when a change request was switched to the status "Resolved". These resolution values are now deprecated since they do not indicate that the change request had really been resolved

```
<metric>
    ...
  <valueCalculators>
     <countEvents id="TransitionsToReopened">
        <event>
           <and>
              <transition field="resolution">
                 <from>FIXED</from>
                 <from>INVALID</from>
                 <from>WONTFIX</from>
                 <from>DUPLICATE</from>
                 <from>WORKSFORME</from>
                 <from>MOVED</from>
                 <from>NOT_ECLIPSE</from>
              </transition>
              <stateFilter>
                 <value field="status">REOPENED</value>
              </stateFilter>
           </and>
        </event>
     </countEvents>

     <countEvents id="TransitionsToResolved">
        <event>
           <and>
              <transition field="resolution">
                 <to>FIXED</to>
                       ...
                 <to>NOT_ECLIPSE</to>
              </transition>
              <stateFilter>
                 <value field="status">RESOLVED</value>
              </stateFilter>
           </and>
        </event>
     </countEvents>

  </valueCalculators>

  <groupEvaluations>
    <calculation name="ProportionOfRework">
      <divide>
        <count valueCalculator="TransitionsToReopened"/>
        <count valueCalculator="TransitionsToResolved"/>
      </divide>
    </calculation>
  </groupEvaluations>
    ...
</metric>
```

All possible resolutions, except „Later“ and „Remind“.

All possible resolutions, except „Later“ and „Remind“.

Fig. 5. Metric Specification of “Number of transitions to ‘Reopened’ divided by the number of transitions to ‘Resolved’ in a time period”

[16]. Instead such change requests should be marked either by setting a target milestone named "Future", by adding the "needinfo" keyword (which means asking more information from the reporter), or by decreasing their priority.

When counting status transitions to "Reopened" it must therefore be distinguished between change requests that had the resolution "Later" or "Remind", and those that had a proper resolution. Only transitions that had a proper resolution can be counted as reopened change requests. Otherwise the resulting values would be distorted for projects that once had used the "Later" and "Remind" resolution.

The **quality indicator** has to be defined in a way that the resulting values can be compared between different projects. The total number of transitions to the "Reopened" status in a certain time period will depend on the size of the project. In order to normalize the result we can divide by the total number of change requests resolved in that time period.

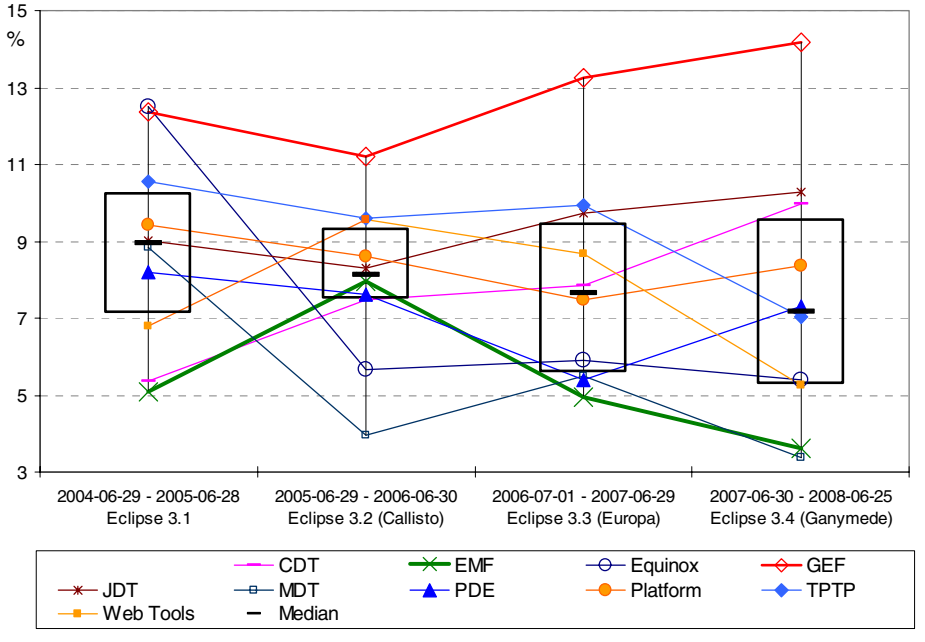


Fig. 6. Percentage of transitions to "Reopened" relative to the number of transitions to "Resolved" in a time period

More precisely we have to decide whether to count resolved change requests in that time period only once, or to count each state transition to "Resolved" of the same change request. Since the numerator (total number of transitions to the "Reopened" status) refers to all incorrect resolutions of a change request, we choose the second option for the denominator, since this corresponds to all resolutions of a change request. Again, state transitions to the "Resolved" status with the resolution "Later" or "Remind" should not be considered, since these change requests have not really been resolved. The corresponding metric specification is shown in figure 5.

Figure 6 shows the resulting measurement values of a number of large projects. Again it can be stated that the EMF project performs better than the average project, while GMF had a high proportion of reopened change requests. Additionally it can be stated from our experience that the GMF project has provided few major new features in the Europe and Ganymede release, and concentrated more on fixing defects.

5 Conclusion and Outlook

In this paper we presented a procedure for developing and validating metric definitions based on CRM data that can be used to evaluate quality characteristics of the development process. It is based on bidirectional quality models which provide an approach how to relate high-level quality characteristics to the technical view of measurable quality properties.

Summarizing, the following steps are performed for the development of a metric:

1. Deriving of process **quality characteristics** from the objectives of the organization.
2. Improvement goal based identification of corresponding **quality properties** and initial validation based on the inspection of individual change request lifecycles.
3. Definition of **quality indicators** that enable comparability between projects.
4. Interpretation based on empirical data.

The usage of metric specifications provided by the BugzillaMetrics tool facilitates an iterative refinement of the related metric definitions in steps 2 to 4. Further on the presented procedure guides the validation of underlying assumptions on different levels: by inspecting the lifecycle of individual change requests and by checking whether the results on the project and the project portfolio level match with experience. This enables to uncover wrong assumptions early during development of the metric.

In the presented examples is a one-to-one relation between the quality characteristic and the quality indicator. In general there can be several quality indicators that need to be weighted according to their influence on a quality characteristic.

It depends on the available CRM data which process characteristics can be evaluated. The Eclipse CRM database enables e.g. to consider characteristics like stability of the prospected target milestones, resolution speed of problem reports and enhancement requests, frequency of high-severity bugs, or the stability of the prioritization of change requests.

CRM systems with a more fine-grained workflow definition and more data collected like estimated and actual work time enable the evaluation of a wider range of quality characteristics.

Since BugzillaMetrics can automatically adapt to custom information collected for the change requests in the Bugzilla database, it can straightforwardly be used for related analyses. As an example the orthogonal defect classification (ODC) requires to classify each reported defect according to a *defect type* and a *defect trigger* in order to compare their distribution to an expected distribution in a certain phase of the process [17]. Given

that these classifications are collected in Bugzilla, the corresponding distributions and their change over time can directly be evaluated using BugzillaMetrics.

Furthermore it can be of interest to associate some kind of size metric to the change requests. By end of 2008 an extension of BugzillaMetrics will be released that enables collecting metrics from version control systems by considering the code changes related to a change request. This enables to consider size metrics like the size of a code change in the evaluations. A prerequisite for these evaluations will be the integration of Bugzilla with version control systems, such as CVS and Subversion, as provided by the Scmbug project [18].

Acknowledgements. We would like to thank Kisters AG, Aachen for supporting this work, and the Eclipse Foundation for providing access to the CRM database.

References

1. Cook, J.E., Votta, L.G., Wolf, A.L.: Cost-Effective Analysis of In-Place Software Processes. IEEE Trans. on Software Engineering 24(8) (1998)
2. Grammel, L., Schackmann, H., Lichter, H.: BugzillaMetrics - Design of an adaptable tool for evaluating user-defined metric specifications on change requests. In: Büren, Bundschuh, Dumke (eds.) Tagungsband des DASMA Software Metrik Kongresses MetriKon 2007. Shaker Verlag, Aachen (2007)
3. BugzillaMetrics, `http://www.bugzillametrics.org`
4. Gall, H.C., Lanza, M.: Software evolution: analysis and visualization. In: Proc.of the 28th international Conference on Software Engineering (ICSE 2006). ACM, New York (2006)
5. Canfora, G., Cerulo, L.: Impact Analysis by Mining Software and Change Request Repositories. In: Proc. of the 11th IEEE International Software Metrics Symposium – METRICS 2005, Como, Italy. IEEE CS Press, Los Alamitos (2005)
6. Kagdi, H.H., Collard, M.L., Maletic, J.I.: A survey and taxonomy of approaches for mining software repositories in the context of software evolution. Journal of Software Maintenance 19(2), 77–131 (2007)
7. Sliwerski, J., Zimmermann, T., Zeller, A.: When do changes induce fixes? In: 2nd International Workshop on Mining Software Repositories (MSR 2005). ACM Press, New York (2005)
8. Koponen, T.: RaSOSS - Remote Analysis System for Open Source Software. In: The International Conference on Software Engineering Advances (ICSEA 2006). IEEE Press, Los Alamitos (2006)
9. Gasser, L., Ripoche, G.: Distributed Collective Practices and F/OSS Problem Management: Perspective and Methods. In: Conference on Cooperation, Innovation & Technology (CITE 2003), Troyes, France (2003)
10. ISO/IEC 15939. Systems and software engineering – Measurement Process. International Organization for Standardization – ISO, Geneva (2007)
11. Simon, F., Seng, O., Mohaupt, T.: Code-Quality Management. Dpunkt-Verlag, Heidelberg (2006)
12. Ebert, C., Dumke, R.: Software Measurement: Establish - Extract - Evaluate - Execute. Springer, Berlin (2007)
13. Gamma, E.: Agile, Open Source, Distributed, and On-Time – Inside the Eclipse Development Process. Keynote Talk, ICSE, St. Louis, Missouri (2005)

14. Basili, V., Caldiera, G., Rombach, H.D.: The Goal Question Metric Paradigm. In: Encyclopedia of Software Engineering, John Wiley & Sons, Chichester (1994)
15. ViPER - Visual Tooling Platform for Model-Based Engineering, http://www.viper.sc
16. Eclipse Bugs – Remove LATER and REMIND resolutions, https://bugs.eclipse.org/178923
17. Chillarege, R., Bhandari, I.S., Chaar, J.K., Halliday, M.J., Moebus, D.S., Ray, B.K., Wong, M.: Orthogonal Defect Classification - A Concept for In-Process Measurements. IEEE Trans. Software Eng. 18(11), 943–956 (1992)
18. Makris, K., Ryu, K.D.: Scmbug: policy-based integration of software configuration management with bug-tracking. In: USENIX Annual Technical Conference. USENIX Association, Berkeley (2005)

Assessment of Business Process Modeling Tools under Consideration of Business Process Management Activities

Andreas Schmietendorf

Berlin School of Economics (FHW Berlin)
Neue Bahnhofstr. 11-17, 10245 Berlin
schmiete@fhw-berlin.de

Abstract. The selection of a business process modeling tool is not new, but nowadays much more complicated than in the past. The reasons for this lie in the necessary consideration of a whole business process management (BPM) approach. After a short introduction, this paper gives an overview about the diverse aspects of BPM-activities and provides an analysis of available evaluation approaches for business process modeling tools. Furthermore, we want to concentrate on an empirical analysis of available modeling tools. This evaluation was executed under consideration of the requirements for a BPM-approach. The kind of investigated tools consider open source tools as well as commercial tool approaches.

Keywords: Business process, management, modeling, evaluation, notations, tools, empirical analysis.

1 Introduction and Motivation

The fulfilment of services as well as the sales of products for an internal or external customer takes place on the basis of business processes. Therefore, business processes define necessary activities to fulfil corresponding tasks. Such activities can be executed with the help of technical systems or with personnel resources (business roles), too. The design of business processes must consider the quantitative and qualitative behavior of activities. Classical business process modeling approaches consider the identification, the documentation and the optimization of selected activities. Beside, there was a strong consideration of organizational aspects.

The business process modeling can be described as image of the reality. By a simplifying view, statements about the qualitative, resource-referential and temporal behavior of a concrete business process should be won. Business process models are required as basis for the implementation of software solutions, too. Especially in the case of modern integration architectures, the consideration of business process models is a must. In the case of service oriented architectures, the interactions of business services should be derived from process models. In this context, it is also spoken of service orchestration as well as service choreography.

This paper investigates the selection procedure of tools for business process modeling. The selection of such kind of tools was mostly discussed under consideration of

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 141–154, 2008.

analysis activities. However, this concerns only one part of the various aspects of the business process management. In many cases an exclusively analytic look at the business processes is only able to comprehend the realities for a short time period and therefore, do not suffice to the complex tasks of the information management any longer.

Nowadays process models are used as fundament for the identification, optimization, implementation, monitoring and management of the whole added value chain inside an enterprise; hence, a selection approach should consider the whole lifecycle of process- and information modeling activities.

The following statement emphasizes the importance of the selection of a business process modeling tool (Source [Blechar 2007]):

> "Business process analysis tools are key components of business process improvement and business process management initiatives. Process modeling is a key capability of a business process management suite."

Another motivation for the use of a business process modeling tool can be found in [Erl 2005]:

> "The advent of business process management has resulted in an industry-wide flurry of process modeling and remodeling activity. Process models have therefore become a central form of business analysis documentation in many organizations."

2 Aspects of Business Process Management

The successful management of business processes (GP) is considered as key to the management of lining up challenges. The causes lie particular in the increasing costing and innovative pressure, the continual consideration of new requirements and the business-process-conformal adjustment of assigned information systems.

The business process management considers much more activities than the well known process analysis. Nowadays we can observe a closer relationship between the different IT-related aspects. The vision is to drive the integration of services from the business perspective by the use of process and object models. The focus is on the so called "IT Business Alignment". A model driven approach should bridge the typical gap between the analysis and the implementation of business processes.

In detail, the following tasks can be identified within modern Business Process Management approaches:

- *Business process analysis* – Deals with the identification of existing business processes and the definition of core process areas. These goals can be attained with the evaluation of available process documentation, examinations within interviews or with the use of specific analysis workshops. Success depends thereby considerably on a pragmatic approach.
- *Business process modeling* - This task takes place with the help of graphic notations. In the industrial area we can observe the use of event driven process chains (EPC), the application of activity diagrams from the Unified Modeling Language (UML) or nowadays the consideration of the Business Process Modeling Notation (BPMN). The used notation should be as comprehensible as possible for all involved.

- *Business process optimization* – Business process models allow the identification of redundancy process activities, an improved synchronization behavior or the identification of process gaps. Based on these findings, it is possible to propose necessary organizational changes as well as the definition of new requirements for existing or new information systems.
- *Simulation of business processes* – Under the use of the established process models, instances of business processes are examined within a dynamic analysis. This involves the so called "what-if analysis" and/or the investigation of synchronization problems, too. The application of actuality-near model variables is important to a successful simulation.
- *Automation and integration* – The direct derivation of implementation assets (e.g. XML) is part of this task. According to this, graphical representations of business processes are transferred into corresponding process description languages. By the use of business services it is therefore possible to establish assembled (orchestrated or choreographed) software applications.
- *Business process monitoring* - The monitoring and control of business processes implies the collection of measurements from executed real process instances. As a result, a large covering is required between target and actual process. The description of this activity is very often described as Business Activity Monitoring (in short BAM).

Under consideration of these tasks the business process analysis or the business process modeling can not be evaluated in particular; thus, they rather have to be integrated in the context of this comprehensive setting of tasks for the business process management. Therefore, the selection of a suitable tool for the process modeling is a significant basis for the entire business process management.

3 Evaluation Approaches for BPM-Tools

This section investigates selected evaluation approaches for business process modeling tools. First, we start with an ISO-standard procedure, usable for any kind of software product evaluation. After this we analyze academic research works, as well as typical procedures used within the industries. Furthermore, the current evaluation approaches will be analyzed from the software measurement point of view.

3.1 The ISO/IEC 14598-Standard

The ISO/IEC 14598 series of international standards (currently replaced by ISO/IEC 2504n - Quality Evaluation Division of the ISO/IEC SQuaRE standards) is concerned with the process of software product evaluation, seen from different viewpoints. [ISO 14598]

Part 5 of this standard describes the evaluator's view of the evaluation process and the activities needed to perform an independent software evaluation in terms of quality characteristic as defined in ISO/IEC 9126. ISO/IEC 14598-5 standard does not prescribe any specific quality model; nevertheless, its strong relationship to the ISO/IEC 9126 quality model provides the possibility to use this standard as orientation for an evaluation approach of business process modeling tools.

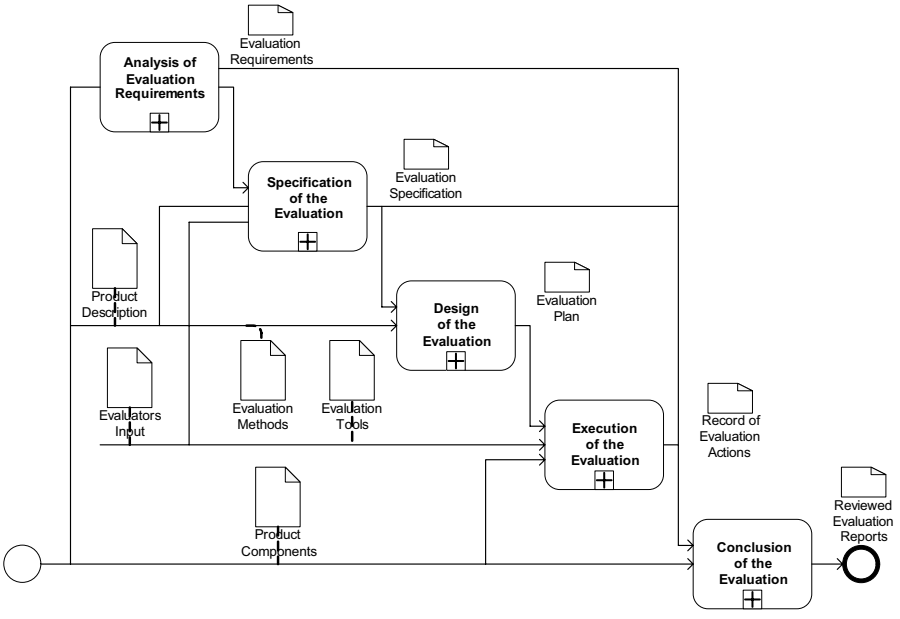


Fig. 1. The Evaluation Process of ISO 14598 (according to BPMN)

3.2 Related Works – Academic Background

[Nüttgens 2002] proposes an evaluation framework with five main categories. The main categories contain further sub categories with approximately 350 separate evaluation characteristics in total. This product independent approach is usable for vendors and buyers and supports the selection of modeling tools for BPM activities. The following overview shows the main categories and selected examples of sub aspects:

- Product & price model
  - License models
  - Service and support
- Manufacturers and customer basis
  - Market and financial strengths
  - Customer references
- Technology & interfaces
  - Language support
  - Provides interfaces
- Methodology & Modeling
  - Provided model notations
  - Management of developed models
- Applications & integration
  - Animation and simulation possibilities
  - Support of measurements

The application of the evaluation approach is supported by tools. Furthermore, it is possible to concentrate on selected evaluation perspectives.

Other approaches use important questions to derive the requirements for the choice of a tool. The following questions are typical examples (under consideration of [Wimmer 2006]):

- Which goal positions are pursued with a modeling approach? (e.g. identification of the current situation including process gaps, simulation possibilities, process costs)
- Which contents should be modeled? (e.g. data/information, activities, organizations, process flows, technical and human resources)
- Who are the users of the tool? (e.g. business analyst, system engineer, reengineers, process designers, application developers)
- Who should be able to read the produced models? (e.g. customers, users, process designers, application developers)
- Where is modeling incorporated within the project cycle? (e.g. strategy development, system analysis, requirements definition, process performance measurement)
- Which scope is modeled? (e.g. selected aspects of a whole organization, core business models for large enterprises)
- Are interfaces available to other systems or applications? (Differentiation of pure indication tools or support in several phases and several levels)
- Which costs are necessary? (e.g. license models, required software and hardware, maintenance costs)
- Which are the required skills in order to use the tool? (e.g. available skills inside the enterprise, offered learning systems)

With the help of questions it should be possible to reflect the specific requirements for a tool selection. The answers of a question can be used to derive measurements, typical with an ordinal scale behavior. Besides, the measured values can be weighted, or be marked as compelling requirement.

3.3 Related Works – Industrial Background

Gartner use the so called Magic Quadrant as graphical representation of a specific marketplace within a defined timeframe. This research tool allows the classification of vendor's product offerings. The four quadrant fields contain the following classification for the analyzed vendors (Source: [Blechar 2007]):

- *Challengers* are well established vendors who have less completeness of vision than the leaders. But they are able to drive a development on basis of their economic strength.
- *Leaders* provide products with a high degree of functionality and have a strong market penetration. They are also well-positioned for the future (vision and investment).
- *Visionaries* are differentiated by innovation. Such innovation can be derived from technological areas or sales and marketing aspects.
- *Niche Players* offer usually products with special characteristics. Therefore, these kinds of vendors concentrate mostly on corresponding market gaps.

The evaluation of the vendors and their products occurs by using the following criteria definitions (Source: [Blechar 2007]):

Ability to Execute
- Products & Services (e.g. product/service capabilities, quality, feature sets and skills)
- Overall Viability (e.g. financial health, practical success of the business unit)
- Sales Execution and Pricing (e.g. capabilities to support pre-sales activities)
- Market Responsiveness and Track Record (e.g. agility to consider customer needs)
- Marketing Execution (e.g. influence the market → mind share)
- Customer Experiences (e.g. customer support programs)
- Operations (e.g. maturity of the organizational structure)

Completeness of Vision
- Market Understanding (e.g. understanding of buyers requirements)
- Marketing Strategy (e.g. consistent set of messages within Web sites, advertising)
- Sales Strategy (e.g. kind of sales channels)
- Offering Strategy (e.g. kind of product development)
- Business Model (e.g. logic of underlying business propositions)
- Vertical/Industry Strategy (e.g. strategy to direct resources, skills and offerings)
- Innovation (e.g. consolidation activities)
- Geographic Strategy (e.g. work with partners, channels and subsidiaries)

The presented criteria are applied with different weightings.

Figure 2 shows an example of a Magic Quadrant for Business Process Analysis tools. It shows IDS Scheer as leading vendor with the highest ability to execute, as well as the provider with the matured vision. Gartner emphasizes that the evaluation result is not a purchase recommendation! (see also [Blechar 2007]).

Further commercial evaluation approaches are provided by Forrester Research. A specific evaluation for Business Process Modeling tools can be found in [Peyret 2006]. The evaluation takes place through the analysis of the current product, the strategic orientation and the market presence of the supplier. He assessed 16 tools of different suppliers and used 121 criterions for it. Analyzed tools are classified by the use of "Risky Bets", "Contenders", "Strong Performers" and "Leaders".

3.4 Related Works – Public Background

A relatively simple approach from the German government can be found in [CCVBPO 2005]. Only 21 criteria were considered for the evaluation. The guideline deals with the following main characteristics:

- Functionality (e.g. architectural aspects, modeling and reporting possibilities)
- Costs (e.g. software licenses, hardware requirements)
- Efforts for staff trainings (e.g. skill requirements, offered trainings)
- Expanded functionalities (e.g. supported interfaces, modeling methodology)

Beside the prepared criteria catalog, the guideline explains further BPM-related aspects. There is a description of possible tasks during the business process analysis/

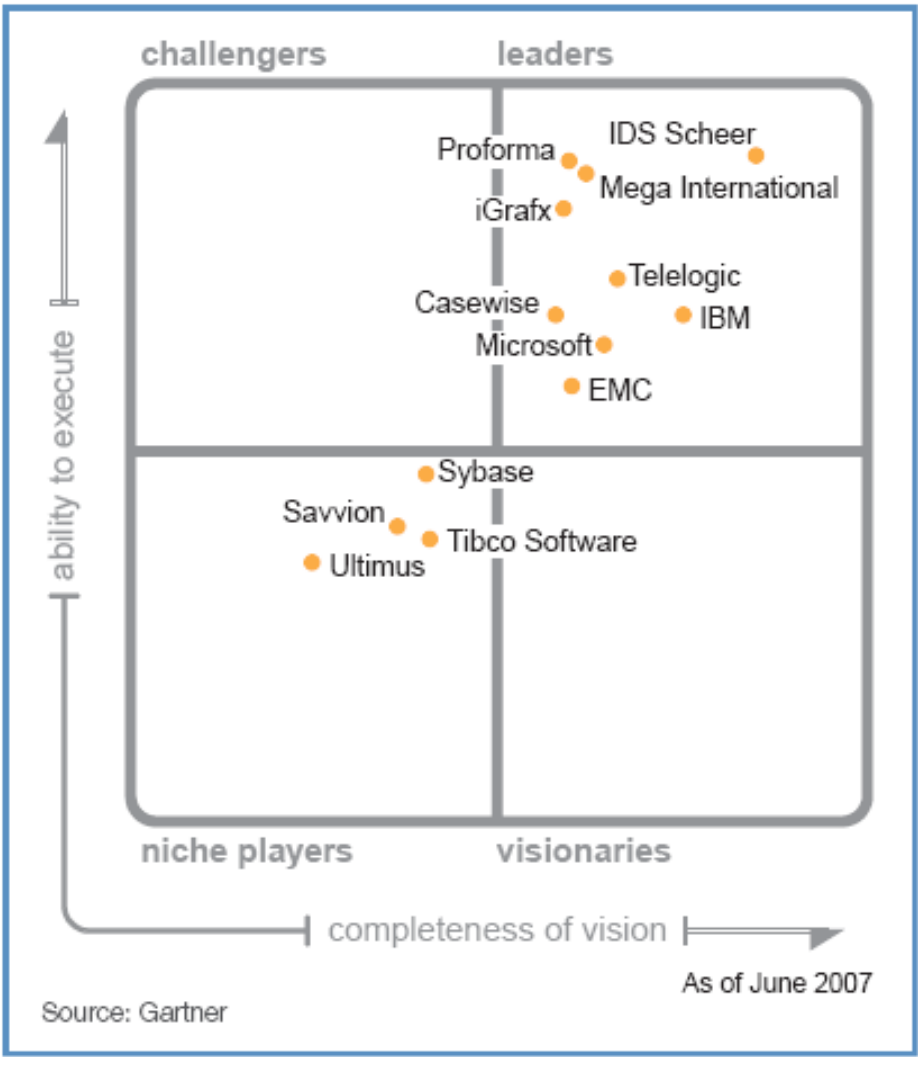


Fig. 2. Magic Quadrant for business process analysis tools (Source: [Blechar 2007])

optimization. Furthermore, the guideline contains a list of reachable benefits within BPM-projects and the peculiarities of a BPM project within the public sector.

3.5 Evaluation of the Current Situation

Figure 3 compares the explained evaluation approaches under consideration of 5 aspects. These aspects are the following ones:

- *BPM covering* deals with the consideration of the different aspects within a whole business process management approach, like mentioned in section 2.

- *Evaluation scope* considers the width of the used evaluation criterions from a functional point of view. (product-, process- and resource-related aspects)
- *Process orientation* means the consideration of necessary steps to carry out a product evaluation. (e.g. under consideration of the ISO 14598 standard)
- *Strategy Strength* deals with evaluation aspects to characterize the strategical orientation of product approaches, for instance an available roadmap.
- *Business Strength* means such aspects like financial health of a possible vendor or the transparency of provided license models.

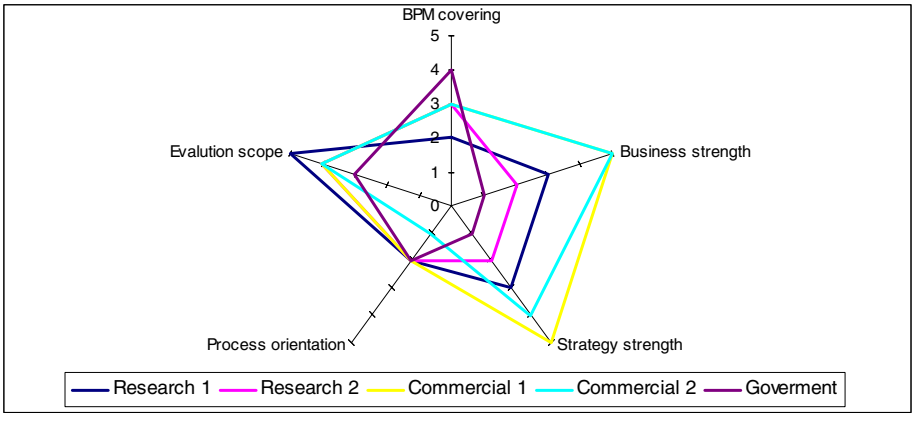


Fig. 3. Comparison of evaluation approaches

For each aspect we used an ordinal scale to characterize the fulfillment degree.

0 – No consideration
1 – Weak mentioned
2 – Selected aspects
3 – Mentioned
4 – Mentioned with explanations
5 – Complete complaint

3.6 Summarizing Remarks

The investigated evaluation approaches support a product decision, but an adoption is necessary in any case. Such an adaptation has to take the specific use conditions and goals into account. Examples for such aspects are explained in the following:

- *Functional conditions* like the required modeling and analyzing functionalities. Another aspect could be the functional restrictions from the used IT-infrastructure (e.g. operating and network systems or required interface technologies)
- *Resource-referential conditions*, like available skills from the employees, performance-related restrictions or economic restrictions. In addition, appropriate resources are required to fulfill management- and maintenance-related aspects.

- *Process-referential conditions*, like the enterprise wide application. For this kind of use an appropriate standard should be defined. A modeling guideline might/shall be helpful for naming conditions, abstraction possibilities or process interfaces.

Furthermore, an evaluation process with a description of necessary steps is required. The investigated evaluation approaches have shown weaknesses here, as displayed in figure 3. The ISO 14598 can help to identify the needed steps.

The multiplicity of a matured evaluation approach is demonstrated in figure 4. An evaluation approach has to consider:

- Customer specific requirements - should be derived by the aims of an enterprise specific BPM-approach. Therefore, the application of the GQM-method can be recommended (see [Solingen 1999]).
- State of the art requirements – can be evaluated by the use of external evaluation, like the mentioned Gartner-Reports. Furthermore, these requirements can be supported by guidelines (e.g. enterprise related).

A catalog of evaluation criteria can help to find out important requirements. In addition, we recommend the use of measurements as formal result descriptions.

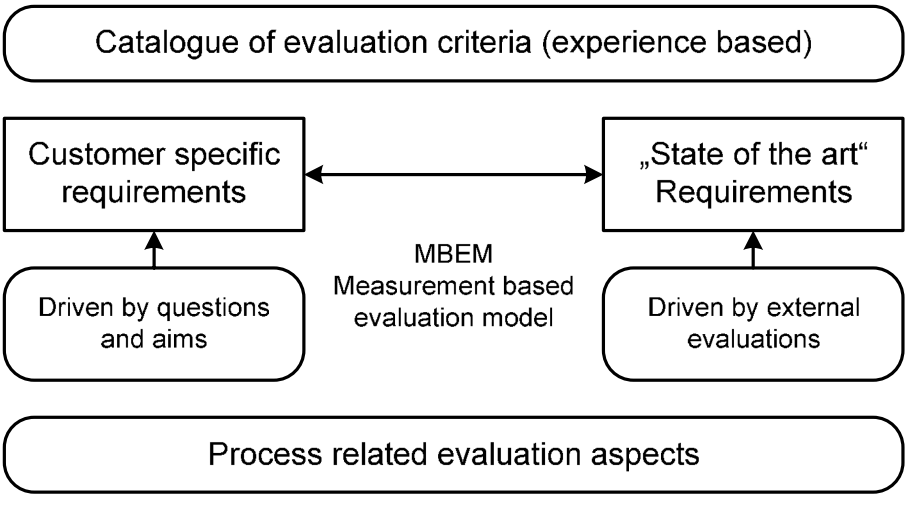


Fig. 4. Multiplicity aspects of an evaluation approach

4 Empirical Analysis of BPM-Tools

The following excerpt shows selected aspects of an empirical investigation. The analysis considered 41 tools, applicable for business process management related aspects. A first analysis was executed in June 2007 under consideration of 36 tools. Both analyses were produced with the help of [BPM 2008]. For the analysis, the following aspects were taken into account:

- Degree of the market relevance
- Supported tasks of BPM-aspects

- Supported modeling notations
- Supported interface formats
- Supported report functionalities

The shown results concentrate on the analysis from September 2008. A comparison of the results with those of the last year occurs only for selected aspects.

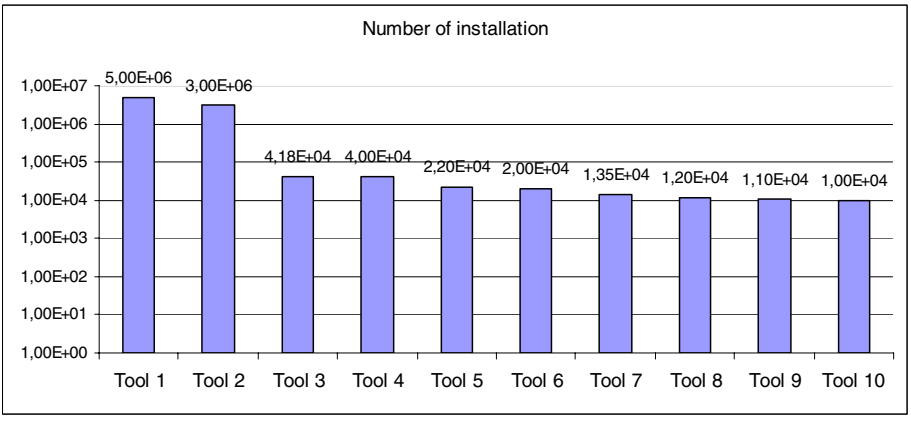


Fig. 5. Number of installations (date of the analysis: September 2008)

Figure 5 shows the number of installations for the most significant tools. It must be considered that the y-axis has a logarithmic scale. This information was available for 61% of all listed tools under [BPM 2008]. The vendor with the most installations provides products and services since 1987. Please pay attention to the fact that this information was not available for the ARIS-framework.

As shown in figure 6 the primary usage of the analyzed tools deals with modeling aspects. The support of simulation possibilities and enabling model driven approaches for software development aspects is not sufficient. But business process management goals, as mentioned in section 2, require the consideration of the whole lifecycle of the information function. This result corresponds to the experiences from [Molle 2007]. As he pointed out, currently available BPM-systems separate the process-logic from the running applications. With other words, an integrated view on business processes and information systems is still missing.

The supported modeling notations can be seen in figure 7 (for a short description of these notations sees Appendix A). During the last year the BPMN-support was improved from 44% to 51% (first analysis from June 2007). The application of the EPC-notation is especially promoted within the German speaking community. The UML support is currently very specific. Typical is that the tools only support selected UML-diagrams and not the whole standard.

Furthermore, the possibilities to generate source code assets or XML-based process descriptions are insufficient, as shown in figure 8. A native XML-support can be identified as state of the art, even though the support of a business process description language is required to close the gap between the modeling and implementation point

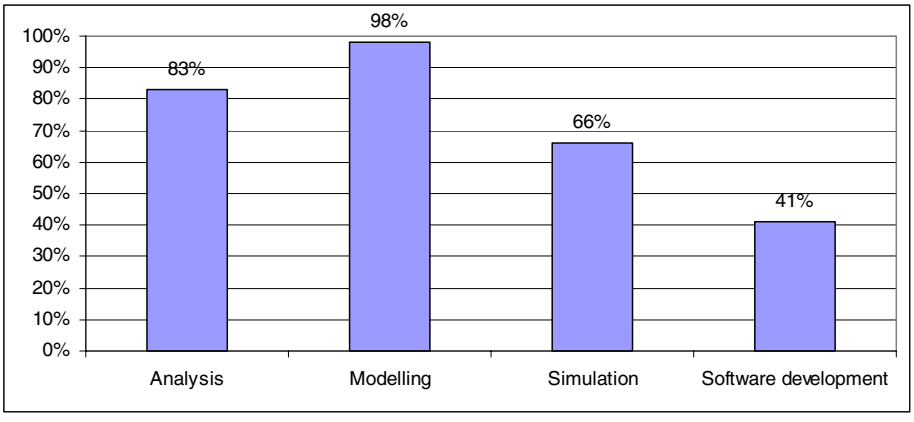


Fig. 6. Usage areas of the tools (date of the analysis: September 2008)

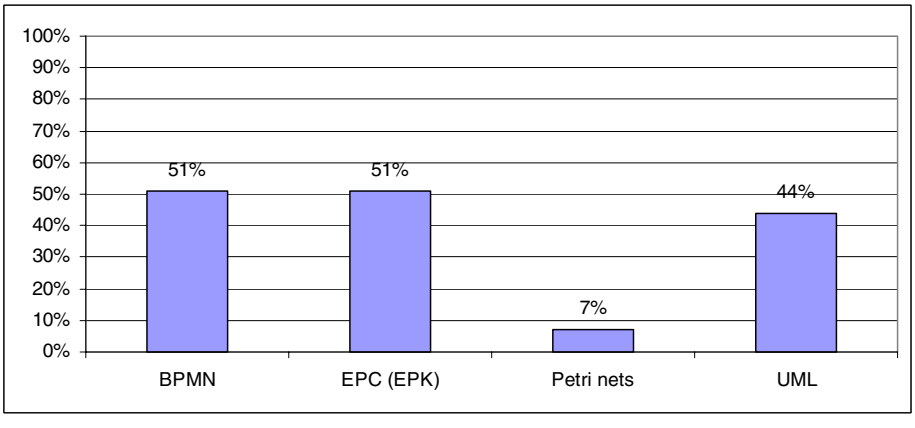


Fig. 7. Supported notations (date of the analysis: September 2008)

of view. For example, the support of BPEL (Business Process Execution Language) allows the controlling of implemented business services under consideration of modeled business processes. Only this way fulfills the requirements in the context of the so called "IT business alignment".

A web based report is an important feature for business process modeling tools. 90% of all analyzed tools support this kind of report. Until now, there are only few tools with an animated representation support. This functionality would be helpful for the visualization of process simulations. Mostly, such functions are only available within the modeling tool itself.

The shown results consider only selected criteria; therefore, a complete evaluation is impossible. With the help of these results it is possible to see trends in BPM-products and currently not covered functionalities. In every case, the effort of the

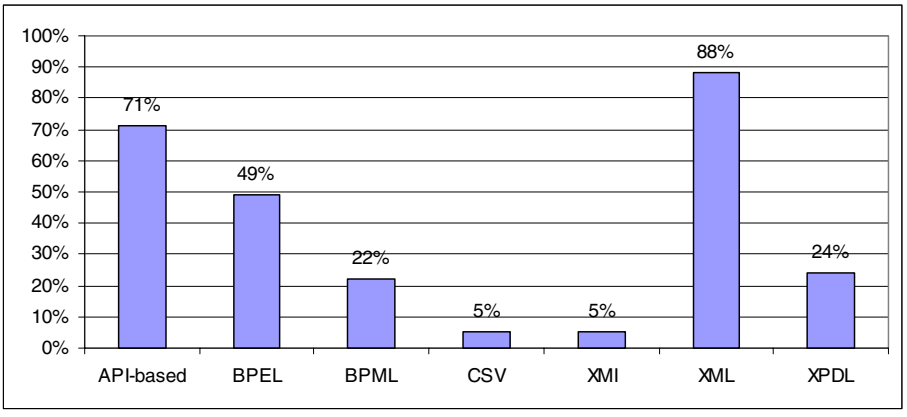


Fig. 8. Supported interface formats (date of the analysis: September 2008)

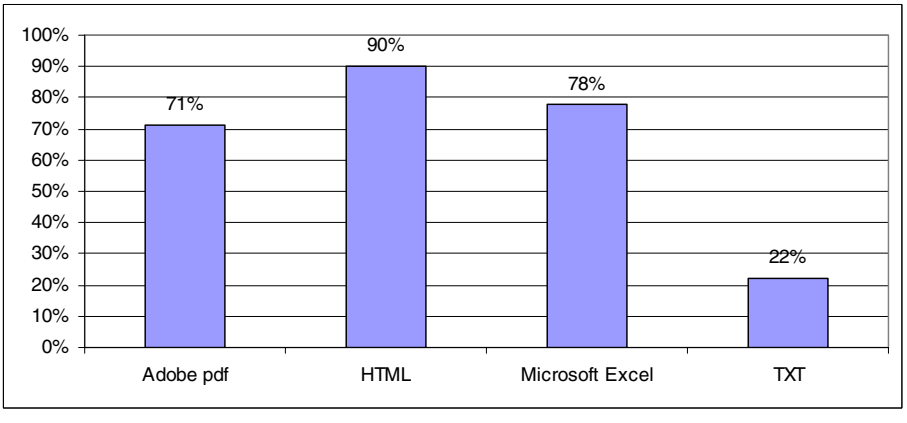


Fig. 9. Supported formats for reports (date of the analysis: September 2008)

manufacturers can be observed regarding an integral support of the whole BPM-activities. The support of the BPMN- and UML-notation can be identified as an important trend, so that every business process modeling tool must support these standard approaches.

5 Conclusions

Within this paper we investigated evaluation approaches for business process modeling tools. The evaluation perspective was driven from a business process management standpoint. As shown there it is not a holistic evaluation approach. Therefore, we can assume that a specific evaluation must be adopted under the use of identified requirements. The in the framework discussed influence criteria can help to establish such an evaluation approach. An aspect which was not investigated within this paper is the

use of prototypical tests. Such a test can help to validate product statements, to check integration possibilities or to evaluate the necessary effort to solve real problem situations. In order to keep the costs low for such tests, a pre-selection should be executed on the basis of the here introduced evaluation aspects.

References

[Blechar 2007] Blechar, M.J.: Magic Quadrant for Business Process Analysis Tools, Gartner RAS Core Research Note G00148777 (June 2007)

[BPM 2008] BPM-Netzwerk.de – Das D.A.CH.-Netzwerk für BPM Professionals, http://www.bpm-netzwerk.de/intro/startIntro.do

[BPMN 2006] Business Process Modeling Notation (BPMN) - Specification Final Adopted Specification, Object Management Group (February 2006)

[CCVBPO 2005] Softwareprodukte zur Geschäftsprozessanalyse und –optimierung, Kompetenzzentrum Vorgangsbearbeitung, Prozesse und Organisation (CC VBPO), Version 2.1, Bundesverwaltungsamt (June 2005)

[Erl 2007] Erl, T.: SOA Principles of Service Design. Prentice Hall, Upper Saddle River (2007)

[Gadatsch 2007] Gadatsch, A.: Grundkurs Geschäftsprozessmanagement: Methoden und Werkzeuge für die IT-Praxis. Vieweg-Verlag (2007)

[ISO 14598] ISO/IEC DIS 14598-5 - Information Technology - Evaluation of software product - Part 5: Process for evaluators, Centre for Software Engineering, Dublin City University, http://www.cse.dcu.ie/essiscope/sm4/14598-5.html

[Nüttgens 2002] Nüttgens, M.: Rahmenkonzept zur Evaluierung von Modellierungswerkzeugen, in Gesellschaft für Informatik (GI) e.V (Hrsg.), Rundbrief der GI-Fachgruppe WI-MobIS, 9. Jahrgang, Heft 2 (November 2002)

[Oestereich 2003] Oestereich, B., Weiss, C., Weilkiens, T., Schröder, C., Lenhard, A.: Objektorientierte Geschäftsprozessmodellierung mit der UML. dpunkt Verlag, Heidelberg (2003)

[Peyret 2006] Peyret, H., Teubner, C., Moore, C., Kim, E., Fossner, L.: The Forrester Wave™: Business Process Modeling Tools, Q3 2006, Forrester Research (2006)

[Solingen 1999] Solingen, V.R., Berghout, E.: The Goal/Question/Metric Method. Mc-Graw Hill Verlag (1999)

[Wimmer 2006] Wimmer, M.: Vorlesung Modellierung betrieblicher Informations systeme II (MoBIS II), Universität Koblenz Landau (2006)

Appendix A – Modeling Notation

Under consideration of [Gadatsch 2007] we can classify notations for the business process modeling as script based- and graphic-oriented approaches. Script-based approaches allow a very exact process description. Currently we find mostly XML-based languages like BPML (Business Process Modeling Language) or BPEL (Business Process Execution Language) for these one. The following description provides an overview about graphic-oriented modeling approaches (only industrial important approaches were taken into account).

- Activity diagram of the UML-notation
  The Unified Modeling Language (short UML) is primarily used during the object oriented development of a software system. The UML 2.0 provides 13 different

kinds of diagrams to model the structural and dynamical behavior of a software system. Nowadays we can observe the use of specific UML-diagrams for modeling business process aspects, too. The modeling of these aspects can be supported through the Use Case-, Sequence- and Activity-Diagram.
Further information can be found in [Oestereich 2003].

- Business process modeling notation
  BPMN provides a standardized approach for the modeling of business processes. Originally, this notation was developed by the Business Process Management Initiative (BPMI). Since 2005 the Object Management Group is responsible for the further development of BPMN. The approach provides a large set of notation elements to model a business flow.

  The BPMN approach is constrained to support only such modeling concepts that are applicable to business processes. Other modeling aspects like organizational structures, data and information models or business rules are out of scope. Further information can be found in [BPMN 2006].
- Event driven process chain
  The application of event driven process chains (short EPC) was introduced through the ARIS-framework. ARIS stands for "architecture of integrated information systems" and provides a large modeling framework. EPC offers a semiformal approach for the graphical modeling of business process. The basic elements are events, functions and links. The extended form allows the consideration of organizational aspects, used information objects or the structured representation of large business models. Further information can be found in [Scheer 2004].

Used Sources:

[BPMN 2006] Business Process Modeling Notation (BPMN) - Specification Final Adopted Specification, Object Management Group (February 2006)

[Gadatsch 2007] Gadatsch, A.: Grundkurs Geschäftsprozessmanagement: Methoden und Werkzeuge für die IT-Praxis. Vieweg-Verlag (2007)

[Oestereich 2003] Oestereich, B., Weiss, C., Weilkiens, T., Schröder, C., Lenhard, A.: Objektorientierte Geschäftsprozessmodellierung mit der UML. dpunkt Verlag, Heidelberg (2003)

[Scheer 2004] Scheer, A. W.; ARIS - Vom Geschäftsprozess zum Anwendungssystem, Springer Berlin 2004

The Impact of Individual Assumptions on Functional Size Measurement

Oktay Turetken*, Ozden Ozcan Top, Baris Ozkan, and Onur Demirors

Informatics Institute,
Middle East Technical University,
06531, Ankara, Turkey
{oktay,ozden,bozkan,demirors}@ii.metu.edu.tr

Abstract. Having been improved, evolved and standardized by the Organization for Standardization (ISO), Functional Size Measurement (FSM) methods have become widely used. However, the measurers still face difficulties in measuring the software products which include unconventional components. We faced the challenge to observe if different interpretations or assumptions of the measurers cause significant differences in the measurement results. In this study, we present the results of a multiple case study we conducted in order to observe the impact of individual assumptions for well known FSM methods.

Keywords: Functional Size Measurement, COSMIC FSM, IFPUG FPA, MkII FPA.

1 Introduction

Function Point measure has gained considerable interest since it has been first introduced by Allan Albrecht in 1979 [1]. Many variations of this original measure and the method have been developed since then [7], [34].

During the last decade the measurement rules of well known Functional Size Measurement methods have been redefined as a series of ISO standards [15]-[20] and a number of examples have been made available [36] to cover diverse fields of applications. Although Functional Size Measurement (FSM) methods have well written measurement manuals and guidelines were established about the fundamental concepts of the measurement methods; the measurers still face challenges during the implementation of FSM methods for unconventional applications. A manifestation of this challenge would be the amount of required subjective judgment of individuals for measurement.

To observe the impact of assumptions of individuals on the measurement results, we conducted a series of case studies. We applied three well known FSM methods; the International Function Point Users' Group (IFPUG) Function Point Analysis (FPA) [13], Mark II (MkII) FPA [21] and the Common Software Measurement

* This study is supported by The Scientific and Technological Research Council of Turkey (TUBITAK).

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 155–169, 2008.

International Consortium (COSMIC) FSM [35] to an unconventional software product with different expert groups. The three separate teams, composed of two experts each, have utilized the same Software Requirements Specification (SRS) document to measure the functional size of the software with a different FSM method. IFPUG FPA, COSMIC FSM and Mk II FPA methods are chosen since they are the most frequently utilized methodologies by the industry, they have well written measurement guidelines and they all are approved by ISO as international standards.

The case product –KAMA- is a graphical modeling tool. We have chosen KAMA as the case product for its potential to highlight the problems that may arise during the measurement process. It can be described as a challenging application from the measurement perspective, due to the challenges it provides for the identification of the boundaries of its transactions and of data entities.

In this paper, we present the multiple case study results by focusing on the difficulties of applying specific methods and the magnitude of the subjective judgment required by each method. We also discuss the impact of the results and provide our suggestions for minimizing the divergence.

The paper is organized in four sections. The related research is summarized in the second section. The case study is presented in the third section. Finally, the results and the comparison of the cases are given in the fourth section.

2 Related Research

The first FSM method "Function Point Analysis" (FPA) which measures the size by means of "the amount of functionality" was introduced by Allan Albrecht in 1979 [1].

Over the years several methods have been developed to measure the size of the software from the same approach; with the aim of improving the original FPA method and extending the previous methods' applicability in different domains [7], [34].

In 1986, the International Function Point Users' Group (IFPUG) was set up as the design authority of Albrecht's FPA method. Since then, IFPUG has been clarifying FP counting rules and expanded the original description of Albrecht [1]. A number of official IFPUG Counting Practices Manual versions are published [8]-[13].

Mk II FPA method was developed by Charles Symons in 1988 to solve the shortcomings of the regular FPA method [32]. Currently, the Metrics Practices Committee (MPC) of the UK Software Metrics Association is the design authority of the method.

The Netherlands Software Metrics Users Association (NESMA) published the first version of Definitions and Counting Guidelines for the Application of Function Point Analysis in 1990 [27]. This method is also based on the principles of the IFPUG FPA method. The difference is that this guideline gives more concrete guidelines, hints and examples.

COSMIC FSM method was published by Common Software Measurement International Consortium (COSMIC) in November 1999 [2]. This group has been established to develop this new method as the one which would measure the functional size of software for not only business information systems, but real-time systems and hybrids of both.

Due to the proliferation of the methods, a workgroup was initiated by International Organization for Standardization (ISO) in 1996, with the purposes of identifying the fundamental concepts of a measurement process, clarifying the conceptual basis and establishing an international standard for functional size measurement. ISO/IEC joint committee first published ISO/IEC 14143/1:1998 International Standard which defines the fundamental concepts of FSM methods [15]. In the following years, other ISO/IEC standards covering issues of "conformity evaluation of software size measurement methods to ISO/IEC 14143-1:1998", "Verification of Functional Size Measurement Methods", "Reference model", and "Determination of functional domains for use with functional size measurement" have been released [16]-[20].

Currently, IFPUG FPA [22], MkII FPA [21], COSMIC FSM [23], NESMA FSM [24] and FISMA FSM [25] are accepted as international standards for functional size measurement. All these FSM methods measure the functional size of a software product; however, they use different metrics and rules during the measurement process [7].

There exist a number of studies on the evaluation and comparison of the FSM methods. Lother and Dumke [26] evaluated FSM methods with respect to a number of criteria and discussed the issues of FSM such as the suitability of the methods for functional domains and the impact of new technologies. Fetcke et al. [4] proposed a model as a generalized representation of different FSM methods based on the similarities and differences. Rule [30] discussed the similarities and differences between IFPUG FPA and MkII FPA in his study to assist practitioners understand the common principles and objectives of these methods. Rollo [29] evaluated IFPUG FPA, MkII FPA and COSMIC FSM by applying them to a sample e-commerce application and discussed the problems associated with sizing web applications. Gencel et al. [5] presented the results obtained by applying MkII FPA and COSMIC FSM to a real-time software system. In this study, the similarities and the differences between the measurement processes of these methods and the difficulties faced during the measurement are discussed.

Nevertheless, none of the above studies were made on the identification of the individual impacts on functional size measurement. However, further research is needed to observe the impact of individual assumptions on functional sizes for well known FSM methods when applied to non-conventional cases. Our prior work [37] focuses on the effect of different interpretations of a specific concept –entity generalizations- on measuring the functional size with IFPUG and COSMIC methods. This study provides a broader view to the implications of individual assumptions on the concept, also by extending the case study to include the application of the MkII method.

3 Case Study

We conducted a multiple case study in order to identify the cases when measurers should make subjective judgments during functional size measurement and to evaluate the measurement processes of different methods in specific cases.

The case study involved the measurement of the functional size of a graphical modeling tool by applying IFPUG FPA [13], MkII FPA [21] and COSMIC FSM [35]

methods. Description of the case project and the case study conduct are discussed in the following sub-sections.

3.1 The Case

KAMA is a modeling tool developed to provide a unified conceptual modeling approach to respond to the needs of the Turkish Armed Forces for modeling and simulation projects. The tool establishes a common modeling approach by supporting a specific notation based on the Unified Modeling Language (UML) [28] and provides a repository to be shared among developers of the system.

The project was started in June 2005 and completed in July 2007. The project staff consisted of 1 project manager, 1 assistant project manager, 2 steering committee members, 1 project coordinator, 8 researchers, 1 software development team leader, 1 quality assurance team leader, 4 software engineers (1 part-time), 1 part-time test engineer and 2 quality engineers (1 part-time). In total, 21 project staff participated in the project. The total effort utilized for the project is 1,832 person-days. The details of the efforts utilized related with the life cycle phases of the project are given in Table 1.

In the project, Rational Software Architect was used to depict software analysis and design; Requisite Pro was used as the requirements management tool; and C# was used as the programming language. Related IEEE standards were utilized for the project work products, which were kept under configuration control by the Subversion tool.

Table 1. The development effort for the case project

Software Development Life Cycle Phase	**Effort (person-days)**
Development Processes	1,287
Software Requirements Analysis	227
Software Design	185
Software Coding & Unit Testing	670
Testing	205
Management	135
Supporting Processes	410
Total	**1,832**

With respect to CHAR Method defined in [19], the functional domain of the KAMA is determined as 'Information System'.

KAMA supports the development of conceptual models with a set of diagrams, model elements and their relationships. Each diagram consists of a specific set of model elements and relationships between them. Briefly, the notation comprises 8 diagram types, 10 model element types and 15 relationship types. The diagram entity has a common set of attributes maintained for all types. For the model elements and relationship entities, on the other hand, together with the common attributes that are maintained by all, there exist attributes specific to types. A partial data model showing the model element, relationship and diagram entities are given in Fig. 1.

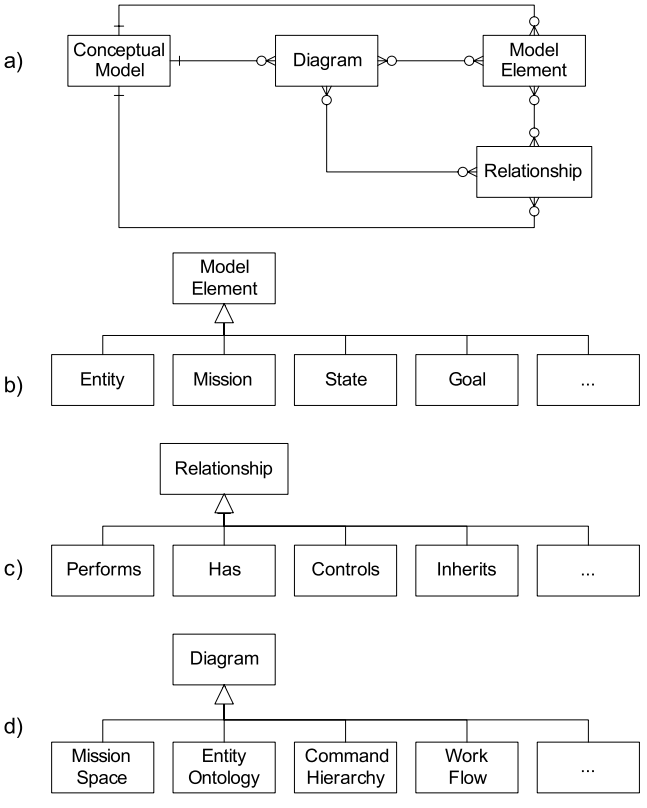


Fig. 1. Entity generalizations for diagrams, model elements and relationships

In KAMA, the user is able to create a new conceptual model of a project, and then create diagrams as a part of that conceptual model. The user can also create model elements such as 'actor', 'role', 'mission' and the relationships between these elements (see Fig 1). Examples for the functional user requirements of KAMA are as follows:

- Create a new Conceptual Model
- Create a Diagram
- Create a Model Element
- Create a new Mission Space Diagram
- Create a new Actor Model Element
- Create a new Task Model Element
- Create a new Actor Model Element on a Mission Space Diagram
- Create a new Task Model Element on a Mission Space Diagram
- Associate a specific Actor Model Element with Mission Space Diagram

3.2 Case Study Conduct

We established three separate teams of experts to measure the functional size of the same software product independently. Each team consisted of two measurers and applied MkII FPA, IFPUG FPA and COSMIC FSM, independently.

Each measurer utilized the same SRS document to measure the functional size of the software. SRS document included requirements statements, UML use case diagrams, activity diagrams as detailed description of the use cases and simple class diagrams as the data model.

Then, the measurers cross-checked the measurement catalogues to bring into light the measurement variances caused by the assumptions and interpretations of different measurers. Each group's measurement result was reviewed and verified with respect to the rules of each method by a member of the other team, who himself was not involved in that measurement process. For example, the measurer who applied MkII FPA reviewed the IFPUG FPA measurement results performed by another expert.

As for the experience of the measurers, two of them holds PhD degree in the related subjects and have considerable practical experience in functional size measurement. Three PhD. students and one MSc. student are doing their thesis work on functional size measurement. All of them received training for at least two of the FSM methods mentioned above and they have previously measured at least one project. However, none of them is certified measurers for any of the methods.

Measurement by IFPUG FPA. The Base Functional Components (BFC)1 of IFPUG FPA is 'Elementary Processes'. In IFPUG FPA, the functional size comprises two aspects: Data and Transactional functions [13]. These involve different BFC Types2. A Data function can be an Internal Logical File (ILF) or External Interface File (EIF). A file is a user identifiable group of logically related data or control information. ILFs are maintained within the boundary of the application whereas EIFs are maintained within the boundary of another application. The functional complexity of each file is based on the number of record element types (RETs) (subgroup within a file) and the number of data element types (DETs) within a file. A DET is a unique user recognizable, non-repeated field (entity attribute). The complexity of a file can be low, average or high, each corresponding to an IFPUG function point value.

The functional size measurement with IFPUG v4.2.1 was performed by two measurers; one performed and the other cross-checked. The total effort utilized is 102 person-hours. The measurement results are given in Table 2.

Table 2. Case Project - IFPUG FPA Size Measurement Details

No. of Elementary Processes	No.of ILFs	No.of EIFs	No.of EIs	No.of EOs	No.of EQs	Functional Size (IFPUG FP)
45	11	0	26	1	18	306

Discussion of the Results. IFPUG FPA measurement process does not give any precedence rule for identifying the data and transactional functions. However, in our case study we started with the data functions since determining the logical files is helpful in identifying the transactional functions and valuing their functional

[1] BFC: "an elementary unit of FUR defined by and used by an FSM Method for measurement purposes" 15.

[2] BFC Type: "A defined category of BFCs. A BFC is classified as one and only one BFC Type" 15.

complexity. In IFPUG FPA the complexity of a transactional function is dependent on the number of ILFs/EIFs maintained during the transactions as well as the total number of input and output DETs.

The IFPUG FPA takes the complete inheritance hierarchy as a single *file*, with a record element type (RET) for each subclass [13]. RETs are optional or mandatory *subgroup* of data elements within an ILF or EIF. They influence the degree of complexity (low, average, high) for *files*. For example, in our case, the complete inheritance hierarchy of 'model elements' is considered as one ILF with a number of RETs for each special type (Fig 2). Thus, with respect to the counting rules, the functional complexity of the "model element" ILF is *high* and so the contribution on the total functional size is 15 IFPUG function points (FP). The affect of number of RETs on functional size were limited in the sense that, with 10 RETs for each of the special "model element type" having attributes of their own, the contribution of the ILF is increased from 7 to 15 function points (complexity level from low to high).

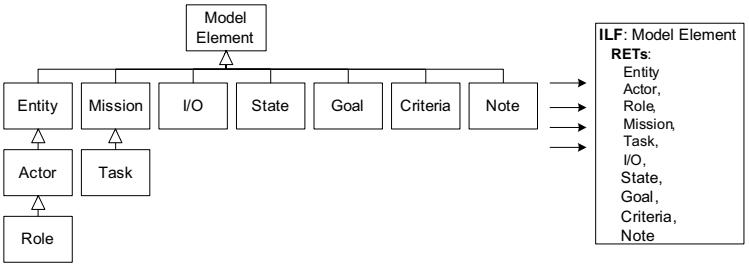


Fig. 2. A mapping from entities to an ILF in IFPUG FPA

Unifying all special entities of an inheritance hierarchy into an ILF also combined many of the transactions performed on each of the special entity. For example, a transaction of creating an 'actor' model element was combined with creating a 'state' model element, even though system may need to behave in a different way for each of them. It can be argued that those two entities (actor & state model elements) are separate in the user domain and whether the application handles both entities in the same way or not can be a design choice rather than a decision to be given in the requirements phase. For applications similar to KAMA, where entity abstractions (aggregation, generalization, etc) are applied extensively, the difference for those two cases can be significant. For example, the functional size of the elementary process of creating a "model element" is 6 FP (functional complexity level being high). Alternatively, having separate model element creation process for each special type would result significantly larger values in total. For 10 specific types, the result would be 60 FP (each having 6 FP with functional complexity level high). If the same practice for other generalized entities (relationship and diagram types) and related elementary processes (update, deletion, read, etc.) are applied, the difference would be more substantial.

Based on these assumptions, where we consider each special type as a separate ILF in the user domain, we re-measured the size and the resulting value turned out to be 1789 FP, as opposed to 306 FP in the first measurement performed in the case study.

The number of elementary processes increased from 45 to 260 and the number of ILFs increased from 11 to 41. 485% difference in the functional size is significant.

Another notable difficulty about IFPUG FPA is related to the counting rules for transactional functions. In IFPUG, one of the following rules is applied in order for an elementary process to be counted as a unique occurrence of an elementary process (external input-EI, external output-EO or external inquiry-EQ) [13]:

"The set of data elements identified is different from the sets identified for other external inputs/outputs/inquiries for the application."

This rule can raise concerns in the context of entity generalization/specialization. Because the rule can be interpreted in a way which is different than the practices applied in the counting manual and other guiding sources [6]. For example, with respect to the practices applied regarding the rules in the counting manual, creating an 'actor' and 'state' model elements is considered as a unique external input maintaining the 'model element' ILF. However, with respect to the rule given above, we can argue that, if the 'actor' and 'state' model elements have different attributes other than the ones they have in common, creating each of them can be considered as different elementary processes. Because, creating an 'actor' model element will maintain a different set of DETs than creating the 'state' model element. This interpretation yet again may result considerable differences in the result. In order to observe the affect of such an interpretation on our case project, we recalculated the functional size. The resulting size value was 512 FP, which is 67% more than the original 306 FP value. The number of ILFs remained the same but the number of elementary processes increased from 45 to 82. Hence, different interpretations and assumptions regarding the counting rules and the structure of the data leaded to differences in functional size, which was significant for our case.

Measurement by MKII FPA. Mk II FPA aims to measure the information processing amount and uses the functional user requirements (FURs) to measure the functional size. The BFCs of this method are the Logical Transactions (LTs). There are no categories of BFCs, i.e. there is only one type of BFC; the LT. The LTs are identified by decomposing each FUR into its elementary components. Each LT has three constituents; input, process and output components.

The base counts are derived by counting Input Data Element Types (DETs) for the input component, by counting the Data Entity Types Referenced for the process component, and by counting the Output DETs for the output component. The functionality involved in providing each of these three distinct kinds of information processing is different. Therefore, the functional size of each LT is computed by multiplying the size of each component by a weight factor which are calibrated to industry-average relative effort to analyze, design, program, test and implement these components in order to enable these three kinds of functionality to be combined into a single value for a Functional Size. Then, the functional size of each LT is summed up to compute the functional size of the whole system.

The functional size measurement by MkII FPA was performed by two measurers; and another measurer checked the results. The total effort utilized is 125.6 person-hours. The measurement results are given in Table 3.

Table 3. Case Project – MkII FPA Size Measurement Details

No.of Logical Transactions	No.of Input DETs	No.of Output DETs	No.of References to Data Entity Types	Functional Size (MkII FP)
283	2.423	3.151	986	3.861,36

Discussion of the Results. When measuring the product by MkII FPA, we had difficulty particularly in identifying Functional Processes and Data Entity Types.

The LT in MkII FPA is defined as "the lowest level of self-consistent process. It is *triggered* by a *unique event* that is of interest to the user, which, when completed, leaves the application in a *self-consistent state* in relation to the unique event" [21].

In our case, the nested and chained nature of the FURs made it difficult to identify the LTs. For example, as a part of a specific diagram, a new model element might be created. A model element might also be created independently, i.e. not associated to a diagram (see some example FURs in Section 3.1). Therefore, we had to decide which ones are the triggering events and hence the LTs;

- the need to create a specific diagram which triggers creating that diagram and all the model elements as part of it (1 LT) or,
- the need to create a diagram, which triggers creation of a diagram; and the need to create a model element on a diagram, which triggers creation of a model element on a diagram (2 LTs).

Moreover, different kinds of diagrams start with the need to create a new type of diagram. They all have a common processing until a point is reached and then the types of processing differentiate. This issue is strongly related to the identification of Data Entity Types which are read or maintained within a LT. The Data Entities involve sub-groups of data entities, called 'Sub-Entity' in MkII FPA. Two assumptions might result in very different functional size measurement results:

- defining LTs which read or maintain the highest level Data Entities in the inheritance tree; or
- considering separate LTs for reading or maintaining sub-entities.

In our case, we have 8 diagram types and 10 model element types. Therefore, we might have 18 LTs or 2 LTs depending on our assumption. Both are true with respect to the rules for identifying LTs in MkII FPA.

Therefore, although finding the triggering event and looking for a consistent state in relation to that event works good in identifying the LTs for most of the time, MkII FPA requires additional rules for deciding what to take as a LT in such cases discussed above.

Measurement by COSMIC FSM. In COSMIC FSM [18], the BFCs are the 'Functional Processes' (FP) and each of these FPs is assumed to comprise a set of sub-processes, called Data Movement Types. Data movement types are the BFCs of this method. There are four kinds of Data Movement Types: Entry (E), Exit (X), Read (R), and Write (W). Each of these is defined as a BFC Type. A data movement moves one or more data attribute belonging to a single 'data group', where each included data

attribute describes a complementary aspect of the same 'object of interest'. An object of interest is any 'thing' or a conceptual object that is identified from the point of view of the functional user requirements. It is equivalent to 'entity-type' in entity relationship (ER) analysis or 'class' in UML [28]. The functional size value in COSMIC FP is the total number of data movements performed in the software system.

The functional size measurement of KAMA with COSMIC FSM v3.0 method was performed by three measurers. Two measurers performed the measurement and one cross-checked the results. Totally, 105 person-hours of effort were utilized. The measurement results are given in Table 4.

Table 4. Case Project - COSMIC FSM Size Measurement Details

No. of Functional Processes	No.of Entries	No.of Exits	No.of Reads	No.of Writes	Functional Size (COSMIC FP)
55	61	154	314	160	697

Discussion of the Results. During the measurement process, one of the difficulties was the determination of Object of Interests (OOI) and related data groups. The data model given in the SRS document of the case product was in the form of simple Entity/Relationship diagrams, which were insufficient in clarifying the OOIs. OOIs were determined only after the domain knowledge is gained and the software requirements are well-understood. The resulting OOIs and their relations looked much alike the OOIs after following data normalization steps up to 3rd Normal Form. Accordingly, we identified three OOIs; diagram, model element and the model-diagram relationship. Sticking to COSMIC measurement manual [35] and business application guideline [36], OOIs were determined in a consistent way, hence it was observed that COSMIC FSM requires a well-defined specification of requirements.

The major difficulty we faced during measurement was the identification of the functional processes that maintain a set of OOIs that can be abstracted to a general entity. One of the main reasons for that was the lack of clear assistance in the measurement manual for distinguishing such processes. For applications similar to KAMA, where entity abstraction (generalization/specialization, aggregation, etc.) can be widely applied, the decision to select the general entity or the specific one for a functional process can result significant differences on the total functional size.

As an initial tendency, the group considered all special entities as separate OOIs and specified each transaction performed on them as separate functional processes. For example, creating a 'command hierarchy' diagram and 'work flow' diagram are considered as separate functional processes. Because, they maintain entities that can be considered as separate in the user domain; the requirements related to those two diagrams are specified explicitly and discretely in the requirements specification document and they are triggered by events that can be perceived as independent by the functional users (triggering event 1 - the user wants to create an 'command hierarchy' diagram; triggering event 2 - the user wants to create 'work flow' diagram). With these assumptions, the measurers identified 270 functional processes.

However, later, knowing that the processing logic of the functional processes which maintain sub-entities is almost identical, the group considered those functional processes in one functional process, which reads or maintains the OOI at a higher level of abstraction in the inheritance tree. Based on this practice, the group obtained 55 functional processes in total, which was only 20% of the value obtained in the first measurement.

The practice mentioned above was a consequence of the interpretation of specific rules given in COSMIC FSM guideline (for sizing business applications software) [36]. These rules are related to *sizing* functional processes that maintain sub-entities (given that functional processes are identified). The COSMIC FSM measurement manual [35] also recommends the reader to refer to this guideline for the details on determining object of interests and separate data groups. The guideline introduces a new term -'sub-type object of interest'- to better handle generalizations between entities. The general practice is to treat each sub-type as a separate object of interest, if there is a need to distinguish more than one sub-type in the *same* functional process. Hence, according to the rules in the guideline, instead of having separate functional processes for each special entity, their contribution on the functional size was taken into account by including additional data movements for each of the special entity (sub-type object of interest) in the functional processes. However, if the functional process did not need to distinguish special entities, only the general entity is referred. For example, creating a model element is a functional process that requires distinguishing each type of model element. For 10 model elements, there were 10 Entry and 10 Write data movements in the functional process.

Another difficulty is related to the identification of the functional processes that are performed within another functional process; a form of a series of functional processes which we termed as 'nested transactions'. As in many standalone applications, in KAMA data is kept in the memory until a 'save' operation is performed. Save operation transfers data from the volatile memory to a persistent storage (in the form of database records, files, etc.). Only then the data truly becomes persistent and continues to exist beyond the life of the application. Otherwise, it is lost. Having a 'save' feature generally implies giving an opportunity for the user to quit the application without saving changes (discarding changes). Nested transactions may occur during the period between a number of entities are created in the memory and then 'saved' into a persistent storage. For example, during the construction of a diagram, we create the diagram, and then we put model elements on the diagram or associate existing ones with the diagram. We also establish relationships between model elements on the diagram. Each of these activities can be considered as separate functional processes (create diagram, associate model element, create relationship, etc). The user can also perform several other functional processes before he issues a 'save diagram' command and saves the diagram and all other related entities into a persistent storage and ends the functional process. In other words, creation (or update) of a diagram entity is spanned through several other functional processes (creation/update of model elements, associating model elements with diagrams, creating relationships between model elements, etc.). That is, the diagram creation is considered as a nested functional process.

However, an alternative view was raised in the measurement group to treat save operations as separate functional processes. Because, considering the save operation

as a part of the creation; functional process ignores the specific requirements of the user, explicitly stating the need for saving data or quitting the application without saving. This practice also ignores the features for 'save as' and undo-redo, which are closely related. Ignoring this feature implies considering two applications -one having undo-redo, 'quitting without saving data' functionalities and the other one lacking those- as equivalent in terms of COSMIC functional size. With these concerns, the group's initial tendency was to treat save operation as a separate functional process that transfers data from the memory to the database. Hence, creating a diagram and saving it to the database considered distinct functional processes, where save operation was considered to be triggered by a unique event of 'saving the diagram data into the database'. But later on team has gave up treating the "save operation" as a separate functional process.

Assuming reuse among functional processes in the counting process, disregarding the opportunities in the solution domain such as design or code reuse, had a large effect on total functional size of the application.

4 Conclusions

In this paper we reported the case results evaluating the impact of individuals on measurement. Three of the widely used FSM methods have been evaluated from this perspective and the challenges concerning the measurement process are identified.

The functional sizes of the case project were measured to be 306 with IFPUG FPA, 3,861.36 with MkII FPA and 697 with COSMIC FSM. Although the unit of measurement for each FSM method is not the same, the variance of the results is far beyond any conversion formula suggested. When we utilize conversion formula stated by Symons in [33], the size of the case project was measured as 743.58 for MkII FPA. This shows that there is a 419.7 % error between the measured result and the expected one.

$$MkII\_FP = 0{,}9 \times IFPUG\_FP + 0{,}005 \times IFPUG\_FP \quad (1)$$

Desharnais et.al in [3] discusses previous studies on the conversion between IFPUG and COSMIC sizes and proposes a conversion formula that is comparable to the findings of prior works and stay within the same range with few outliers.

$$Y(CFP) = 1{,}0*(IFPUG\_UFP) - 3 \quad (2)$$

When the proposed formula was applied to the findings of this case study, the size of the product with COSMIC was expected to be 303 CFP, with respect to 697 CFP value actually measured. The error ratio is 130%.

Although; each team has utilized the same SRS document, the number of BFCs for the IFPUG FPA, MkII FPA and COSMIC FSM were determined as 45, 283 and 55 respectively. The definitions of the FSM methods for the BFCs are not the same, so these numbers are not supposed to be the same. However, the concepts beyond these definitions are similar to each other. That is, although it is not unusual to have a minor difference in results, the realized amount of divergence cannot be justified solely by the difference of the measurement rules.

The case study results reveal that unconventional cases might subject to different interpretations and assumptions of the measurers during functional size measurement.

We observed that the teams involved in the measurement process made different assumptions for the determination of BFCs and BFC Types, which resulted in significant variance among functional sizes measured by different individuals. The variances are largely due to different assumptions of measurers when measuring the specific cases where 'entity abstractions' (generalization & aggregation) are commonly used and there is considerable "functional similarity" among the functional processes and when the rules to identify the BFCs do not give guidelines for specific cases such as in nested and chained transactions.

The definitions given by IFPUG FPA method allows practitioners to combine the entities which are inherited from a high level entity into an ILF and thus combine their functional processes into one functional process. The selection of a general entity can result in significant differences on the total functional size (see Section 3.2.1). The effects of such interpretations for IFPUG FPA and COSMIC FSM methods are also discussed in [37] in detail.

Although the COSMIC FSM manual and the COSMIC Business Application Guideline give guidelines for a wide range of situations, there is no clear assistance for the entity generalization concept. Measurers have difficulty in identifying the functional processes maintaining a set of OOIs that can be abstracted to a general entity. When the measurers considered each special entity as a separate OOI and specified each transaction performed on them as separate functional processes; the numbers of the functional processes were determined as 270. Later on, the measurement team based their interpretations on the "sub-type OOI" concept given in the COSMIC FSM Guideline. Assuming that the processing logic of the functional processes which maintain sub-entities is almost identical; the measurers combined those functional processes in one functional process and counted 55 functional processes which is only 20% of the previous one.

Although The MkII Measurement Manual does not include an explicit guidance leading the practitioners measure the software product from entity abstraction perspective; measurers are able to make assumptions by the help of the "sub-entity" concept during the measurement of the cases that use entity abstraction. The guideline emphasize that sub-entities should be counted separately for the ones using different processing logic and should be considered as a general entity and counted as one for the ones using the same processing logic.

The 'entity generalization' concept is closely related to the 'functional similarity' concept defined as "similarity between the software functions" by Santillo and Abran [31]. Because, as each sub-entity type is considered as another entity to be read or maintained, then the number of transactions increase, and hence the functional size. In our case study, although their processing logics are similar and their functionality can considerably be reused, we arrived at separate transactions (LTs or Functional Processes) each triggered by different events each maintaining different sub-entities. For example, we might have 18 or 2 transactions depending on whether defining the transactions which read or maintain the highest level Data Entities in the inheritance tree or considering separate transactions for reading or maintaining sub-entities.

Moreover, the nested and chained nature of the functional user requirements made it difficult to identify the triggering events and hence the transactions (see Section 3.2.2). Different assumptions of the measurers resulted in different sizes.

Therefore, the measurement manuals for the FSM methods can be extended to set or clarify rules not only how to handle 'entity generalization' and 'functional similarity' as well as for the identification of transactions when they are in a nested or chained form.

References

1. Albrecht, A.J.: Measuring Application Development Productivity. In: Proc. of the IBM Applications Development Symposium, Monterey, California, pp. 83–92 (1979)
2. Abran: COSMIC FFP 2.0: An Implementation of COSMIC Functional Size Measurement Concepts. In: FESMA 1999, Amsterdam (October 7, 1999)
3. Desharnais, J.-M., Abran, A., Cuadrado, J.: Convertibility of Function Points to COSMIC-FFP: Identification and Analysis of Functional Outliers. In: MENSURA 2006, Conference Proceedings edited by the Publish Service of the University of Cadiz, Cadiz, Spain, November 4-5, 2006, pp. 190–205 (2006), http://www.uca.es/publicaciones
4. Fetcke, T., Abran, A., Dumke, R.: A Generalized Representation for Selected Functional Size Measurement Methods. In: Dumke, R., Abran, A. (eds.) Current Trends in Software Measurement, pp. 1–25. Shaker (2001)
5. Gencel, C., Demirors, O., Yuceer, E.: A Case Study on Using Functional Size Measurement Methods for Real Time Systems. In: Proc. of the 15th. International Workshop on Software Measurement, Montreal, Canada, September 12-14, 2005, pp. 159–178. Shaker-Verlag (2005)
6. Garmus, D., Herron, D.: Function Point Analysis: Measurement Practices for Successful Software Projects. Information Technology Series. Addison-Wesley, Reading (2000)
7. Gencel, C., Demirors, O.: Functional Size Measurement Revisited. ACM Transactions on Software Engineering and Methodology (to be published, July 2008)
8. IFPUG, Counting Practices Manual (CPM), Release Z.0, IFPUG, Westerville, Ohio (1986)
9. IFPUG, CPM, Release 2.0, IFPUG, Westerville, Ohio (1988)
10. IFPUG, Function Point CPM, Release 3.0, IFPUG, Westerville, Ohio (1990)
11. IFPUG, Function Point CPM, Release 4.0, IFPUG, Westerville, Ohio (1994)
12. IFPUG, Function Point CPM, Release. 4.1, IFPUG, Westerville, OH (1999)
13. IFPUG, Function Point Counting Practices Manual, Release 4.2.1 (2005)
14. IEEE Std. 14143.1: Implementation Note for IEEE Adoption of ISO/IEC 14143-1:1998 - Information Technology- Software Measurement- Functional Size Measurement -Part 1: Definition of Concepts (2000)
15. ISO/IEC 14143-1: Information Technology - Software Measurement - Functional Size Measurement - Part 1: Definition of Concepts (1998) (updated, 2007)
16. ISO/IEC 14143-2: Information Technology - Software Measurement - Functional Size Measurement - Part 2: Conformity Evaluation of Software Size Measurement Methods to ISO/IEC 14143-1:1998 (2002)
17. ISO/IEC TR 14143-3: Information Technology - Software Measurement - Functional Size Measurement - Part 3: Verification of Functional Size Measurement Methods (2003)
18. ISO/IEC TR 14143-4: Information Technology - Software Measurement - Functional Size Measurement - Part 4: Reference Model (2002)
19. ISO/IEC TR 14143-5: Information Technology - Software Measurement - Functional Size Measurement - Part 5: Determination of Functional Domains for Use with Functional Size Measurement (2004)
20. ISO/IEC FCD 14143-6: Guide for the Use of ISO/IEC 14143 and related International Standards (2005)
21. ISO/IEC IS 20968:2002: Software Engineering - MK II Function Point Analysis - Counting Practices Manual (2002)

22. ISO/IEC IS 20926:2003: Software Engineering - IFPUG 4.1 Unadjusted Functional Size Measurement Method - Counting Practices Manual (2003)
23. ISO/IEC 19761:2003: Software Engineering - COSMIC-FFP: A Functional Size Measurement Method (2003)
24. ISO/IEC IS 24570:2005: Software Engineering - NESMA functional size measurement method Ver.2.1 - Definitions and counting guidelines for the application of FPA (2005)
25. ISO/IEC IS 29881:2008: Software Engineering - FISMA functional size measurement method Ver.1.1 (2008)
26. Lother, M., Dumke, R.: Points Metrics - Comparison and Analysis. In: International Workshop on Software Measurement (IWSM 2001), Montréal, Québec, pp. 155–172 (2001)
27. NESMA, Definitions and Counting Guidelines for the Application of Function Point Analysis, Version 1.0 (1990)
28. OMG, Unified Modeling Language: Superstructure, Ver.2.0, Formal/05-07-04, Object Management Group (2005)
29. Rollo, T.: Sizing e-Commerce. In: Proc. of the ACOSM 2000 – Australian Conference on Software Measurement, Sydney (2000)
30. Rule, G.: A Comparison of the Mark II and IFPUG Variants of Function Point Analysis (1999), http://www.gifpa.co.uk/library/Papers/Rule/MK2IFPUG.html
31. Santillo, L., Abran, A.: Software Reuse Evaluation Based on Functional Similarity in COSMIC-FFP Size Components. In: Software Measurement European Forum – SMEF 2006, Rome, Italy, May 10-12, 2006, pp. 167–176 (2006)
32. Symons, C.: Function Point Analysis: Difficulties and Improvements. IEEE Transactions on Software Engineering 14(1) (January 1988)
33. Symons, C.: Conversion between IFPUG 4.0 and MkII Function Points, Software Measurement Services Ltd., Version 3.0 (1999)
34. Symons, C.: Come Back Function Point Analysis (Modernized) – All is Forgiven!). In: Proc. of the 4th European Conf. on Software Measurement and ICT Control (FESMA-DASMA 2001), Germany, pp. 413–426 (2001)
35. The Common Software Measurement International Consortium (COSMIC), Functional Size Measurement Method Version 3.0 Measurement Manual (2007)
36. The Common Software Measurement International Consortium (COSMIC): Guideline for Sizing Business Applications Software Using COSMIC-FFP, Version 1.0 (2005)
37. Turetken, O., Demirors, O., Gencel, C., Ozcan Top, O., Ozkan, B.: The Effect of Entity Generalization on Software Functional Sizing: A Case Study. In: Jedlitschka, A., Salo, O. (eds.) PROFES 2008. LNCS, vol. 5089, pp. 105–116. Springer, Heidelberg (2008)

Measurement of Functional Size in Conceptual Models: A Survey of Measurement Procedures Based on COSMIC*

Beatriz Marín, Giovanni Giachetti, and Oscar Pastor

Centro de Investigación en Métodos de Producción de Software,
Universidad Politécnica de Valencia,
Camino de Vera s/n,
46022 Valencia, Spain
{bmarin,ggiachetti,opastor}@pros.upv.es

Abstract. Many functional size measurement procedures have been developed for applying the COSMIC measurement method to particular methods of software production. A subset of these measurement procedures is centered on the measurement of the functional size of the applications from their conceptual models, allowing the generation of indicators in early stages of the development cycle of a software product. This paper presents a survey of these functional size measurement procedures in order to provide a guide for practitioners and researchers. Finally, a general analysis focused on the results obtained in the survey is performed to obtain important lessons that must be considered in the development of correct measurement procedures.

Keywords: Functional Size Measurement, Functional Size Procedures, COSMIC, Conceptual Models.

1 Introduction

Nowadays, it is widely accepted that it is essential to know the functional size of applications in order to successfully apply estimation models, effort models, and budget models [33]. This knowledge will allow the project leader to generate indicators to facilitate project management. To measure the functional size of software applications, four measurement methods have been recognized as standards: IFPUG FPA [22], MK II FPA [23], NESMA FPA [24], and Cosmic FFP [21]. The first three methods are based on the Function Point Analysis proposal [1], which takes into account only the functionality of the system that the human user observes. These FPA-based methods have several limitations for the correct measurement of systems: for instance, they only allow the measurement of Management Information Systems, which excludes the measurement of other types of software (such as real time software); they have units that are hard to understand; they do not consider the

* This work has been developed with the support of MEC under the project SESAMO TIN2007-62894 and co financed by FEDER.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 170–183, 2008.

functionality that allows communication between layers in systems with a layer-based architecture, etc. To overcome the limitations of FPA-based measurement methods, the COSMIC measurement method was defined.

In addition, software production processes have evolved from focusing essentially on the solution space (software product) to focusing on the problem space (conceptual models). The new software production processes are based on MDA (Model Driven Architecture) approaches [35], which allow the generation of the applications by means of model transformations. In these technologies, the conceptual models are a key resource that allows the partial or complete generation of the final software product. Consequently, the measurement of the functional size in the conceptual models allows the project leader to generate indicators in early stages of the development cycle of a software product.

Taking into account this situation, many proposals have been defined to measure the functional size of software applications from conceptual models. The aim of this work is to present a broad survey of the existing literature related to functional size measurement procedures based on COSMIC that can be applied to conceptual models. This survey includes the following proposals: Bévo et al. [7], Jenner [25], Diab et al. [15], Poels [39], Nagano et al. [36], Azzouz et al. [5], Condori-Fernández et al. [12], Habela et al. [19], Grau et al. [18], Levesque et al. [29], and Marín et al. [31].

In this paper, we summarize the proposals based on the COSMIC measurement method according to the following criteria [30]: the version of the measurement method, the context of the proposal, the functional domain (i.e., real time systems, management information systems), the input artifact (i.e., a requirements model, an analysis model, and a design model), the rules to apply the procedure, the instrument to apply the procedure, and the verification of the procedure.

The objectives of this paper are (1) to provide researchers with an overview of the current state of the functional size measurement procedures based on COSMIC and (2) to provide practitioners with information about the functional size measurement procedures that are available.

The rest of the paper is organized as follows: section 2 presents the existing proposals of measurement procedures based on COSMIC that allow the measurement of the functional size from conceptual models. Section 3 presents an overall analysis of the proposals. Finally, section 4 presents some conclusions, highlighting the features that must be considered by the functional size measurement procedures.

2 Functional Size Measurement Procedures

In this section, we present eleven proposals of functional size measurement procedures based on COSMIC. It is important to note that the proposals by Nagano et al. [36], Condori-Fernádez et al. [12], and Marín et al. [31] were correctly defined as measurement procedures. Even though the rest of the proposals presented in this survey were not originally defined as measurement procedures, they do correspond to measurement procedures according to the definition of the International Vocabulary of Basic and General Terms of Metrology [20], which defines a measurement procedure as: *a detailed description of a measurement according to one or more measurement principles and to a given measurement method*.

2.1 Proposal of Bévo et al. (1999)

Bévo et al. [7] perform a mapping between concepts of UML diagrams (use cases, scenarios, and classes) and concepts of COSMIC. A general description of this proposal is presented below:

- **Version of the Measurement Method.** Cosmic-FFP version 2.0 [2]
- **Context of the Proposal.** Unified Modelling Language (UML) version 1.0
- **Functional Domain.** Management information systems.
- **Input Artifact.** Diagrams of use cases, scenarios, and classes.
- **Rules to Apply the Procedure.** The boundary of the system to measure is included in the use case diagram. Each use case corresponds to a functional process. The data movements are represented in the scenarios, which are sequences of interactions that occur within a use case. Each class of the class diagram corresponds to a data group, and the attributes of those classes correspond to the data attributes. Each actor corresponds to a functional user. The triggering events and the layers are not represented with concepts of UML diagrams.
- **Instrument to Apply the Procedure.** A tool named *Metric Xpert* [6].
- **Verification of the Procedure.** The accuracy of the proposal was verified [6]. To perform this verification, five case studies were measured with the Metric Xpert tool. Then, the results were compared with the measures obtained by experts, obtaining differences that fluctuated between 11% and 33%.

2.2 Proposal of Jenner (2001)

Jenner [25] discusses the granularity aspect of the use cases in the proposal by Bévo et al. presented above. For this reason, the general characteristics of the Jenner proposal are very similar to the characteristics of the Bévo et al. proposal.

- **Version of the Measurement Method.** Cosmic-FFP version 2.0 [2]
- **Context of the Proposal.** UML version 1.0
- **Functional Domain.** Management information systems.
- **Input Artifact.** Diagrams of use cases, sequences, and classes.
- **Rules to Apply the Procedure.** Each functional process is represented by a sequence diagram because Jenner considers that sequence diagrams represent an adequate abstraction level of the use cases. The data movements are represented by the interaction messages of the sequence diagrams. This proposal also uses swimlanes to represent the layers of a system.
- **Instrument to Apply the Procedure.** This procedure has a tool [26].
- **Verification of the Procedure.** The proposal has been verified using case studies.

2.3 Proposal of Diab et al. (2001)

Diab et al. [15] present a set of formal rules that allow the measurement of the functional size of real time applications that are specified with Real-Time Object

Oriented Modelling (ROOM) [42]. The ROOM specifications are used by the Rational Rose Real Time (RRRT) tool for the design and specification of real time systems. The general characteristics of this proposal are the following:

- **Version of the Measurement Method.** Cosmic-FFP version 2.0 [2]
- **Context of the Proposal.** The design of an RRRT model might be observed through two different view points: structure and behavior. The structure of an RRRT model is based on three kinds of entities: actors, protocols, and data objects. An actor is an active object that has restricted visibility of and by other actors. A protocol represents a set of messages that can be exchanged among the actors. A data object is the basic unit of the system data. On the other hand, the dynamic part of an RRRT model is specified with a finite state machine for each actor. Each state machine can be defined with states, sub-states, actions, and transitions between the states.
- **Functional Domain.** Real time systems.
- **Input Artifact.** RRRT model (static and dynamic part).
- **Rules to Apply the Procedure.** The boundary of the system to be measured is represented by a set of actors. The layers correspond to a set of actors with the same level of abstraction, which must be selected by the practitioners using their human judgment. Each transition corresponds to a functional process. The data movements are represented by actions and messages. Actors and protocol classes correspond to data groups, and the attributes and variables of these classes correspond to the data attributes.
- **Instrument to Apply the Procedure.** A tool named *μcRose*[16]. This tool implements the measurement procedure that is updated to version 2.2 of Cosmic-FFP [21].
- **Verification of the Procedure.** The rules of the proposal have been verified by experts of COSMIC. In addition, this proposal has been applied to case studies, and the results have been compared with the measures obtained by experts. Finally, the tool assures the repeatability and consistency of the proposal.

2.4 Proposal of Poels (2002)

Poels [39] presents a mapping between concepts of COSMIC and the concepts of the business model and the services model of MERODE [14]. Later, this proposal was extended to allow the measurement of multilayer applications [41], specifying that the business model corresponds to a layer, and the services model corresponds to another layer. The general characteristics of this proposal are the following:

- **Version of the Measurement Method.** Cosmic-FFP version 2.1 [4]
- **Context of the Proposal.** The MERODE development method. This method is based on the MERODE conceptual model, which is comprised of a business model and a services model. The business model is composed by a class diagram, an object-event table, and a state transition diagram. The services model specifies the generation of events by the user and their transmission to the business model.
- **Functional Domain.** Management information systems.

- **Input Artifact.** MERODE model (business and services models).
- **Rules to Apply the Procedure.** Poels defines the rules separately for each model of MERODE. The users of the business model correspond to the services model. The boundary of the business model corresponds to the boundary between the business model and the users. Each functional process of the business model corresponds to a set of class methods over all of the enterprise objects, which are invoked by the occurrence of a type of business event. Each data movement corresponds to each class method that composes a functional process. In the business model, the exit data movements are not represented. The data groups correspond to the classes of the business model. On the other hand, the users of the services model correspond to the user interface model (this model is not specified in the MERODE model). The boundary of the services model corresponds to the boundary between the services model and the users. Each functional process of the services model corresponds to a non-persistent service object that is invoked by an input, output or control service request message or by a business event occurrence (for output object only). Again, each data movement corresponds to each class method that composes a functional process, and all the types of data movements are represented in the services model.
- **Instrument to Apply the Procedure.** Manual application of the procedure.
- **Verification of the Procedure.** This proposal has been validated theoretically [40].

2.5 Proposal of Nagano et al. (2003)

Nagano et al. [36] present a measurement procedure to measure the functional size of real time applications specified using xUML [34]. The general characteristics of this proposal are:

- **Version of the Measurement Method.** Cosmic-FFP version 2.0 [2]
- **Context of the Proposal.** The Shlaer-Mellor development method [43]. This method is an object-oriented method that uses xUML to specify systems.
- **Functional Domain.** Real time systems.
- **Input Artifact.** Classes, state transition, and collaboration diagrams.
- **Rules to Apply the Procedure.** The candidate data groups are attributes and relationships between objects of the class diagram. Also, the parameters of messages and control signals are candidate data groups. The triggering events are identified in the collaboration diagrams, which include the relationship between the external entity and the objects of the system. The functional processes correspond to a sequence of data movements. Finally, the data movements correspond to the actions that an object performs to move it from one state to the next state according to the collaboration diagram.
- **Instrument to Apply the Procedure.** Manual application of the procedure.
- **Verification of the Procedure.** This proposal has been applied to the Rice Cooker case study [13], and the results were compared with the results obtained by experts, obtaining a difference of 53%.

2.6 Proposal of Azzouz et al. (2004)

Azzouz et al. [5] present a tool that automates the measurement of the functional size of applications developed with the Rational Unified Process (RUP) [28]. The general characteristics of this proposal are:

- **Version of the Measurement Method.** Cosmic-FFP version 2.2 [21]
- **Context of the Proposal.** Rational Unified Process. This method uses UML to specify the systems.
- **Functional Domain.** Management information systems.
- **Input Artifact.** Use case diagrams, scenarios, and detailed scenarios.
- **Rules to Apply the Procedure.** Azzouz et al. base their proposal on the rules described by Bévo [7] and Jenner [25]. However, Azzouz considers that the layer cannot be represented in the UML diagrams. Therefore, the user of the tool must manually identify the layers of the system. Also, this proposal adds a stereotype to identify the triggering events in the use case diagrams. The measurement is performed in three phases of RUP: in the business modeling and requirement analysis phase, the artifact used is the use case diagram; in the analysis phase, the artifact used is the scenario; and in the analysis and design phase, the artifact used is the detailed scenario.
- **Instrument to Apply the Procedure.** A tool integrated in the Rational Rose tool.
- **Verification of the Procedure.** The tool was verified using the Rice Cooker case study [13].

2.7 Proposal of Condori-Fernández et al. (2004)

Condori-Fernández et al. [12] present a measurement procedure to estimate the functional size of object-oriented systems from the requirements specifications that are defined using the OO-Method approach [37]. The general characteristics of this proposal are:

- **Version of the Measurement Method.** Cosmic-FFP version 2.2 [21]
- **Context of the Proposal.** The development method OO-Method. This method is based on a formal language. It is an object-oriented method that allows the automatic generation of final applications by means of model transformations [38]. The software production process in OO-Method is represented by three models: the requirements model, the conceptual model, and the execution model. The requirement model specifies the system requirements using a set of techniques such as the mission statement, the functions refinement tree, and the use case diagram. To establish the traceability between the requirements model and the conceptual model, the requirements model uses sequence diagrams. The conceptual model captures the static and dynamic properties of the functional requirements of the system (object, dynamic, and functional models). The conceptual model also allows the specification of the user interfaces in an abstract way through the presentation model. The execution model allows the transition from the problem space to the solution space. The software product can be generated in a systematic and automatic way for different platforms.

- **Functional Domain.** Management information systems.
- **Input Artifact.** OO-Method requirements model (functions refinement tree, use case diagrams, and sequence diagrams).
- **Rules to Apply the Procedure.** The boundary of the system to be measured corresponds to the border between the set of use cases and the actors of the use case diagram. Each functional process corresponds to each elementary function of the functions refinement tree (primary use case). Also, each secondary use case corresponds to a functional process. The data groups are identified in the sequence diagram. Each different actor, control class or entity class of the sequence diagram corresponds to a data group. The data movements correspond to the messages of the sequence diagrams. In this proposal a single layer is identified because there is not a functional partition at the requirements level. The triggering events are not represented.
- **Instrument to Apply the Procedure.** Manual application of the procedure.
- **Verification of the Procedure.** This proposal has been rigorously verified in several ways: according to measurement theory [9]; in conformity with COSMIC [9]; using the formal framework DISTANCE [9]; performing empirical studies of its repeatability and reproducibility [11], and evaluating its adoption in practice [10].

2.8 Proposal of Habela et al. (2005)

Habela et al. [19] present an extension of the use case model that allows the measurement of the functional size using COSMIC. The general characteristics of this proposal are:

- **Version of the Measurement Method.** Cosmic-FFP version 2.2 [21]
- **Context of the Proposal.** UML version 1.5
- **Functional Domain.** Management information systems.
- **Input Artifact.** Use case diagrams, and detailed use cases using a template that includes references to business rules, pre-conditions, post-conditions, and a description in steps of the main and alternatives scenarios.
- **Rules to Apply the Procedure.** Each use case corresponds to one or more functional processes. The data movements are identified in each step described in the scenarios. Each step specifies the movement of a set of data attributes. The uses, extends, and generalizations between use cases are taken into account to avoid redundancies in the measurement.
- **Instrument to apply the Procedure.** Manual application of the procedure.
- **Verification of the Procedure.** We did not find studies of validation, verification, or application of this proposal.

2.9 Proposal of Grau et al. (2007)

Grau et al. [18] present a set of mapping rules to measure the functional size of i* models generated by means of reengineering of systems using PR*i*M [17]. The general characteristics of the Grau et al. proposal are the following:

- **Version of the Measurement Method.** Cosmic-FFP version 2.2 [21]
- **Context of the Proposal.** The PR*i*M method, which is a process reengineering i* method that addresses the specification, analysis and design of information systems from a reengineering point of view. In PR*i*M, the i* model is comprised of two models: an operational i* model (that contains the functionality of the system), and an intentional i* model (that contains the non-functional requirements). To generate the operational i* model, scenario-based templates named *Detailed Interaction Scripts* are used. These templates describe the information of each activity of the current process by means of pre-conditions, post-conditions, triggering events, and a list of actions undertaken in the activity.
- **Functional Domain.** Management information systems.
- **Input Artifact.** Detailed interaction scripts, and an operational i* model.
- **Rules to Apply the Procedure.** The boundary of the system to be measured corresponds to the actor of the operational i* model that represents the different pieces of the system. The users are actors of the operational i* model that represent one or more human roles. The data movements are identified in the operational i* model and correspond to any dependency where the *dependum* is a resource. Each functional process corresponds to an activity of the detailed interaction scripts. The triggering events are part of the conditions associated to the activity. Finally, the data groups correspond to the resources of the detailed interaction script.
- **Instrument to Apply the Procedure.** A tool named *J-PRiM*.
- **Verification of the Procedure.** This proposal has been applied to the C-Registration case study [27], and the results have been compared with the results obtained by experts, obtaining a difference of 53%.

2.10 Proposal of Levesque et al. (2008)

Levesque et al. [29] apply COSMIC to measure the functional size of systems from use case diagrams and sequence diagrams. This proposal classifies the functional processes in two groups: data movement types and data manipulation types. The general characteristics of the Levesque et al. proposal are the following:

- **Version of the Measurement Method.** Cosmic-FFP version 2.1 [3]
- **Context of the Proposal.** UML version 1.4, and UML version 2.0
- **Functional Domain.** Management information systems.
- **Input Artifact.** Use cases and sequence diagrams.
- **Rules to Apply the Procedure.** For the functional processes corresponding to the data movement type, each use case is a functional process. The actors of the use case are the users. The entities of the sequence diagram are the data groups. The data movements correspond to the messages among the entities of the sequence diagram. On the other hand, the data manipulations correspond to the conditions associated to the error messages of the sequence diagrams. Finally, this proposal obtains the functional size aggregating the messages between the actors and objects of the sequence diagrams.
- **Instrument to Apply the Procedure.** Manual application of the procedure.

- **Verification of the Procedure.** This proposal has been applied to the Rice Cooker case study [13], and the results have been compared with the results obtained by experts, obtaining a difference of 8%.

2.11 Proposal of Marín et al. (2008)

Marín et al. [31] present a measurement procedure to measure the functional size of object-oriented systems generated in MDA environments from their conceptual models. This proposal uses the OO-Method development method [38] as the reference MDA approach. The general characteristics of the Marín et al proposal are the following:

- **Version of the Measurement Method.** Cosmic-FFP version 3.0 [4]
- **Context of the Proposal.** The development method OO-Method version 3.8. This method is composed by three models: the requirements model, the conceptual model and the execution model. The last two models of the OO-Method approach has been implemented in a tool named *Olivanova* [8]. This tool allows the specification of systems with a graphical notation in a conceptual model and allows the automatic generation of software products from this conceptual model. The OO-Method conceptual model is comprised of four models: the object model, the functional model, the dynamic model, and the presentation model. The specification of the systems with these four models allows the automatic generation of fully working applications. The OO-Method applications are generated with a three tier architecture: presentation, logic, and database. Each tier of the architecture is associated with the other tiers in a superior/subordinate hierarchical dependency. Therefore, the presentation tier can use the services of the logic tier because the logic tier is beneath the presentation tier in the hierarchy. In the same way, the logic tier can use the services of the database tier because the database tier is beneath the logic tier in the hierarchy. In addition, the OO-Method applications have at least one software component in each tier of the architecture: the client component, the server component, and the database component. The client component has the graphical user interface of the applications. The server component has the business logic of the application. And finally, the database component has the persistence of the application.
- **Functional Domain.** Management information systems.
- **Input Artifact.** OO-Method conceptual model (object, functional, dynamic and presentation).
- **Rules to Apply the Procedure.** This proposal is structured in the three phases of the COSMIC method: the strategy phase, the mapping phase and the measuring phase. With respect to the strategy phase, the scope of the measurement can be determined by the functional processes, the layers, or the whole application. The layers correspond to the hierarchical tiers of the OO-Method applications: the presentation tier, the logic tier, and the database tier. The pieces of software correspond to the software components: the client component, the server component, and the database component. The users are the human users, the client component, and the server

component of the applications. The users are separated from the pieces of software by a boundary. With respect to the mapping phase, the functional processes are groups of functionality that can be directly accessed by the users. These groups of functionality correspond to the interaction units specified in the menu of the presentation model. The data groups correspond to the classes of the object model that participate in the functional processes. The data attributes correspond to the attributes of the classes identified as data groups. With respect to the measuring phase, the data movements correspond to the movements of data groups between the users and the functional processes. This proposal has 69 rules to identify the data movements that can occur in the OO-Method applications. Finally, this proposal has a set of rules to obtain the functional size of each functional process of the application, of each piece of software of the application, and of the whole application.

- **Instrument to Apply the Procedure.** This procedure has a tool [32].
- **Verification of the Procedure.** This proposal has been verified respect its conformity with COSMIC. Also, the tool has been verified using OO-Method case studies, and the results have been compared with the measures obtained by experts. Finally, the tool assures the repeatability and the consistency of the proposal.

3 General Analysis

In this section, we present an overall analysis of the criteria used in the survey presented above.

With respect to the version of the COSMIC measurement method, we observed that four proposals (Bévo, Jenner, Diab, and Nagano) use the 2.0 version, two proposals (Poels and Levesque) use the 2.1 version, four proposals (Azzouz, Condori-Fernández, Habela, and Grau) use the 2.2 version, and one proposal (Marín) uses the 3.0 version. It is important to note that the proposal oby Nagano (which was defined in 2003) uses the 2.0 version in spite of the fact that newer versions of COSMIC already exited in 2003. It is also important to note that the proposal by Levesque (which was defined in 2008) uses the 2.1 version in spite of the fact that the version 3.0 of COSMIC already existed in 2008. Our opinion is that newer versions of COSMIC provide improvements and clarifications that help to better understand the measurement method and to obtain accurate measures. Therefore, we consider that the use of the last version of the method is very important for the correct development of measurement procedures.

With regard to the context of the procedure, five proposals (Bévo, Jenner, Azzouz, Habela, and Levesque) measure UML models, one proposal (Diab) measures RRRT models, one proposal (Poels) measures MERODE models, one proposal (Nagano) measures xUML specifications, two proposals (Condori-Fernández and Marín) measure OO-Method models, and one proposal (Grau) measures i* models. The UML, MERODE, and i* models do not have enough expressivity to specify all the functional requirements of the applications (for instance, these models do not allow the specification of the values assigned to the attributes of the classes, the interaction

units, etc.). The same situation occurs with the OO-Method requirement models. Therefore, the proposals based on these models only estimate the functional size of the applications. On the other hand, the proposals based on the RRRT model, the xUML specification, and the OO-Method conceptual model have enough semantic formalization to specify all the functional requirements, allowing the measurement of the functional size of the applications.

With respect to the functional domain, we observed that only two proposals (Diab, Nagano) have been developed for the domain of real time systems. The remaining nine proposals have been developed for the domain of management information systems. We did not find any measurement procedure proposal for other domains (such as algorithmic systems, geographical systems, ubiquitous systems, etc.), in spite of the fact that the COSMIC measurement method can be applied to any software system domain.

With regard to the input artifact, all the proposals use more than one input artifact. Seven proposals (Bévo, Jenner, Azzouz, Condori-Fernández, Habela, Grau, and Levesque) use input artifacts obtained in the requirements phase, three proposals (Diab, Poels, Nagano) use input artifacts obtained in the analysis phase, and only one proposal (Marín) uses input artifacts obtained in the analysis and design phase.

With respect to the rules to apply the procedure, only two proposals (Condori-Fernández and Marín) perform the design of the measurement procedure, defining the objectives of the procedure, the characterization of the concept to be measured, the mapping with the concepts of COSMIC, and the measurement rules. The remaining nine proposals only define some mappings between the concepts of COSMIC and the concepts of the conceptual models to be measured. The design of a measurement procedure is a key stage in the development of a measurement procedure (correctly abstracting the elements that will be measured), since, otherwise, the procedure may not measure what should be measured according to the specifications of the base measurement method selected. It is also important to keep in mind the direct influence that the design of a measurement procedure has on the application and possible automations of the procedure.

With regard to the instrument to apply the procedure, six proposals (Bévo, Jenner, Diab, Azzouz, Grau, and Marín) have been automated, and five proposals (Poels, Nagano, Condori-Fernández, Habela, and Levesque) must be applied manually. The manual measurement of functional size is generally very time-consuming and could have many precision errors. Therefore, it is very important to automate the measurement procedures to obtain a solution that can be efficiently applied in academic and industrial environments. In addition, a tool that automates the measurement procedures reduces the measurement costs and measurement training, and ensures perfect repeatability of the measures.

Finally, with respect to the verification of the procedure, we observed that only one proposal (Habela) has not been verified in some way. The remaining proposals have been verified using different techniques: using case studies, performing theoretical validations, performing conformity validations, using empirical studies, etc. Thus, it is important to keep in mind that a high quality design of a functional size measurement procedure is not enough to assure the quality of the measures obtained by this procedure. To ensure the quality of the results obtained, it is also essential to verify the developed procedure.

4 Conclusions

This paper provides an extensive summary of the existing proposals of functional size measurement procedures that are based on the COSMIC measurement method and that use the conceptual models as input artifacts to perform the measurement. The main contribution of this work is the presented survey, which provides researchers with an updated overview of the current state of the functional size measurement procedures that are based on COSMIC. This survey also provides practitioners with valuable information about the functional size measurement procedures that are available.

It is important to remark that the measurement procedures presented in this paper have been developed to apply the COSMIC measurement method to the conceptual models in order to obtain the functional size of final applications in early stages of the software development process. Therefore, some of the important lessons taken from this work are: 1) the measurement procedure must be based on the last version of the measurement method; 2) the input artifact used must have enough semantic formalization to allow the specification of all the functional requirements; 3) the design of the measurement procedure must be carried out, clearly defining rules to specify the strategy of the measurement, rules to perform the mapping between the concepts of COSMIC and the concepts of the conceptual models, and rules to identify the data movements and perform the measurement; 4) the automation of the procedure must be carried out to reduce the cost of performing the measurement and to increase the efficiency of the measurement process; and finally, 5) the verification of the procedure must be carried out to assure the quality of the results obtained.

References

1. Albrecht, A.: Measuring Application Development Productivity. In: IBM Applications Development Symposium, pp. 83–92 (1979)
2. Abran, A., Desharnais, J.M., Oligny, S., St-Pierre, D., Symons, C.: COSMIC-FFP Measurement Manual, version 2.0. Software Engineering Management Research Laboratory, Université du Québec à Montréal - UQAM, Canada (1999)
3. Abran, A., Desharnais, J.M., Oligny, S., St-Pierre, D., Symons, C.: COSMIC-FFP Measurement Manual, Version 2.1. The Common Software Measurement International Consortium (2001)
4. Abran, A., Desharnais, J.M., Lesterhuis, A., Londeix, B., Meli, R., Morris, P., Oligny, S., O'Neil, M., Rollo, T., Rule, G., Santillo, L., Symons, C., Toivonen, H.: The COSMIC Functional Size Measurement Method, version 3.0. GELOG web site (2007), `http://www.gelog.etsmtl.ca`
5. Azzouz, S., Abran, A.: A proposed measurement role in the Rational Unified Process (RUP) and its implementation with ISO 19761: COSMIC FFP. In: Software Measurement European Forum 2004, Rome (2004)
6. Bevo, V.: Analyse et Formalisation Ontologique des Procédures de Mesure Associées aux Méthodes de Mesure de la Taille Fonctionnelle des Logiciels: de Nouvelles Perspectives Pour la Mesure. Doctoral thesis, Université du Québec à Montréal - UQAM, Montréal (2005)

7. Bévo, V., Lévesque, G., Abran, A.: Application de la méthode FFP à partir d'une spécification selon la notation UML: compte rendu des premiers essais d'application et questions. In: 9th International Workshop Software Measurement, Lac Supérieur, Canada, pp. 230–242 (1999)
8. CARE Technologies, http://www.care-t.com
9. Condori-Fernández, N.: Un procedimiento de medición de tamaño funcional a partir de especificaciones de requisitos. Doctoral thesis, Universidad Politécnica de Valencia, Valencia (2007)
10. Condori-Fernández, N., Pastor, O.: An Empirical Study on the Likelihood of Adoption in Practice of a Size Measurement Procedure for Requirements Specification. In: 6th International Conference on Quality Software – QSIC, Beijing, pp. 133–140 (2006)
11. Condori-Fernández, N., Pastor, O.: Evaluating the Productivity and Reproducibility of a Measurement Procedure. In: ER Workshops, pp. 352–361 (2006)
12. Condori-Fernández, N., Abrahão, S., Pastor, O.: On the Estimation of Software Functional Size from Requirements Specifications. Journal of Computer Science and Technology 22(3), 358–370 (2007)
13. COSMIC Group: Rice Cooker – Cosmic Group Case Study. École de technologie supérieure, Université du Québec à Montréal - UQAM, Montréal (2003)
14. Dedene, G., Snoeck, M.: M.E.R.O.DE.: A Model-driven Entity-Relationship Object-oriented Development Method. ACM SIGSOFT Software Engineering Notes 19(3), 51–61 (1994)
15. Diab, H., Frappier, M., St-Denis, R.: Formalizing COSMIC-FFP Using ROOM. In: ACS/IEEE International Conference on Computer Systems and Applications, Beirut (2001)
16. Diab, H., Koukane, F., Frappier, M., St-Denis, R.: μcROSE: Automated Measurement of COSMIC-FFP for Rational Rose Real Time. Information and Software Technology 47(3), 151–166 (2005)
17. Grau, G., Franch, X.: ReeF: Defining a Customizable Reengineering Framework. In: Krogstie, J., Opdahl, A.L., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 485–500. Springer, Heidelberg (2007)
18. Grau, G., Franch, X.: Using the PRiM method to Evaluate Requirements Model with COSMIC-FFP. In: Proceedings of the IWSM-MENSURA 2007, Mallorca, pp. 110–120 (2007)
19. Habela, P., Glowacki, E., Serafinski, T., Subieta, K.: Adapting Use Case Model for COSMIC-FFP Based Measurement. In: 15th International Workshop on Software Measurement – IWSM 2005, Montréal, pp. 195–207 (2005)
20. ISO: International vocabulary of basic and general terms in metrology – VIM (2004)
21. ISO/IEC: ISO/IEC 19761, Software Engineering – CFF – A Functional Size Measurement Method (2003)
22. ISO/IEC: ISO/IEC 20926, Software Engineering – IFPUG 4.1 Unadjusted Functional Size Measurement Method – Counting Practices Manual (2003)
23. ISO/IEC: ISO/IEC 20968, Software Engineering – Mk II Function Point Analysis – Counting Practices Manual (2002)
24. ISO/IEC: ISO/IEC 24570, Software Engineering – NESMA Functional Size Measurement Method version 2.1 – Definitions and Counting Guidelines for the application of Function Point Analysis (2005)
25. Jenner, M.S.: COSMIC-FFP and UML: Estimation of the Size of a System Specified in UML – Problems of Granularity. In: 4th European Conference on Software Measurement and ICT Control, Heidelberg, pp. 173–184 (2001)

26. Jenner, M.S.: Automation of Counting of Functional Size Using COSMIC-FFP in UML. In: 12th International Workshop Software Measurement, pp. 43–51 (2002)
27. Khelifi, A., Abran, A., Symons, C., Desharnais, J.M., Machado, F., Jayakumar, J., Leterthuis, A.: The C-Registration System Case Study with ISO 19761 (2003)
28. Kruchten, P.: The Rational Unified Process: An Introduction. Addison-Wesley, Reading (2000)
29. Levesque, G., Bevo, V., Cao, D.T.: Estimating software size with UML models. In: Proceedings of the 2008 C^3S^2E Conference, Montreal, pp. 81–87 (2008)
30. Lother, M., Dumke, R.: Point Metrics-Comparison and Analysis. In: Current Trends in Software Measurement, Aachen, pp. 228–267 (2001)
31. Marín, B., Condori-Fernández, N., Pastor, O., Abran, A.: Measuring the Functional Size of Conceptual Models in a MDA Environment. In: 20th International Conference on Advanced Information Systems Engineering Forum, Montpellier, pp. 33–36 (2008)
32. Marín, B., Giachetti, G., Pastor, O.: Automating the Measurement of Functional Size of Conceptual Models in a MDA Environment. In: Jedlitschka, A., Salo, O. (eds.) PROFES 2008. LNCS, vol. 5089, pp. 215–229. Springer, Heidelberg (2008)
33. Meli, R., Abran, A., Ho Vinh, T., Oligny, S.: On the Applicability of COSMIC-FFP for Measuring Software Throughout its Life Cycle. In: 11th European Software Control and Metrics Conference, Munich (2000)
34. Mellor, S., Balcer, J.: Executable UML: A Foundation for Model-Driven Architecture. Addison Wesley, Reading (2002)
35. Miller, J., Mukerji, J.: MDA Guide Version 1.0.1 (2003)
36. Nagano, S., Ajisaka, T.: Functional metrics using COSMIC-FFP for object-oriented real-time systems. In: 13th International Workshop on Software Measurement, Montreal (2003)
37. Pastor, O., Gómez, J., Insfrán, E., Pelechano, V.: The OO-Method Approach for Information Systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programming. Information Systems 26, 507–534 (2001)
38. Pastor, O., Molina, J.C.: Model-Driven Architecture in Practice: A Software Production Environment Based on Conceptual Modeling. Springer, New York (2007)
39. Poels, G.: A Functional Size Measurement Method for Event-Based Object-Oriented Enterprise Models. In: 4th International Conference on Enterprise Information Systems – ICEIS, Ciudad Real, pp. 667–675 (2002)
40. Poels, G.: Definition and Validation of a COSMIC-FFP Functional Size Measure for Object-Oriented Systems. In: 7th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, Darmstadt (2003)
41. Poels, G.: Functional Size Measurement of Multi-Layer Object-Oriented Conceptual Models. In: 9th International Object-Oriented Information Systems Conference, Geneva, pp. 334–345 (2003)
42. Selic, B., Gullekson, G., Ward, P.T.: Real-time Object Oriented Modelling. Wiley, Chichester (1994)
43. Shlaer, S., Mellor, S.: Object Lifecycles: Modelling the World in States. Yourdon Press, Prentice-Hall (1992)

Evaluation Aspects for a Sustainable Integration of e-Learning within the Software Engineering (Case Study)

Andreas Schmietendorf [1], Steffen Mencke[2], and Gaby Schmietendorf [3]

[1] Berlin School of Economics, FB II, Neue Bahnhofstr. 11-17, 10245 Berlin, Germany
schmiete@fhw-berlin.de
[2] Otto-von-Guericke Universität Magdeburg, FIN/IVS,
Universitätsplatz 2, 39106 Magdeburg, Germany
{schmiete,mencke}@ivs.cs.uni-magdeburg.de
[3] TU Dresden, Fakultät Verkehrswissenschaften "Friedrich List"
Hettnerstraße 3, 01062 Dresden, Germany
gaby@fsr-verkehr.de

Abstract. The implementation of Blended Learning events is a complex task which has to be well considered and well planned in approach and execution. The following paper presents a field report with not only detailed planning aspects but also analysed realization and evaluated success of the further process. The described course was an event in the range of Software Engineering. Therefore the potential benefits are discussed for industrial software development too.

Keywords: e-Learning, e-Teaching, blended learning, Software Engineering.

1 Introduction

Blended Learning is an innovative teaching concept. It describes the combination of online learning aspects and facets of teaching and learning from face to face [ASTD08]. The development of such concepts is a complex process where different variables have to be considered.

For the realization of the following Blended Learning concept, the course, SOA-Data-Management' was chosen. This course is part of the bachelor or master degree programme in Business Informatics at the Hochschule Harz (University of Applied Science). Content of the course is the service-oriented integration of distributed information systems where so called Web Service technologies are in the center of attention. The focus is on the theoretical foundations and on the methodical handling of classical integration problems respectively.

The scope of this course amounted to 60 hours in total within winter semester 2008 (equates to 4 contact hours per week). In addition to this course a supporting web site was available at the university and the communication beside the courses was realized on the base of e-mails until now. The web site was used for 3 years and within its framework lecture notes, exercises and instructions or descriptions of basic conditions of project works were provided.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 184–195, 2008.

By the choice of this course, both the possibilities of a learning platform and the extensively accomplished exercises as well as occurred problems in this context should be tested and documented in a centralized way. The storage of the generated documents should be carried out by the students themselves within a protected member's area. Furthermore, the choice of this particular course was focused because of the low number of attending students. As a result, a manageable communication effort was supposed.

In the next chapter 2, the development of the used Blended Learning concept will be demonstrated. Later on, the implementation of this strategy will be described in chapter 3 and the real execution of the Blended Learning course will be discussed in chapter 4. The evaluation is carried out in chapter 5. Finally, a summary and outlook will complete the paper.

2 The Development of a Blended Learning Concept

In-class lectures should be reserved for mediation or developing technical and methodical skills respectively. A further intention of the course was the establishment of project teams within time of attendance. Moreover, a temporal capacity to deal with the exercises was given. Especially presenting and if necessary discussing the achieved results should be realized during this time. The proving of the deployed online tools in the framework of in-class lectures was an emphasis, too. The aspired advantages for the students were:

- Providing support in knowledge compilation within private study by offering continuative sources of information.
- Supply of a communication platform for cooperative handling of excercices to be undertaken or final project reports respectively.
- Assistance with realtime communication and dicussion of problems as well as the possibility to hand in feeback anonymously.
- Saving the acquired results of exercises and projects by depositing previously created documents on the appendant group sites.
- Simulation of selected elements in distributed software development or service integration respectively (in this case: specifically communication platform).
- Animation of process driven service integration where the BPMN-notation and the process discription language BPEL should explained.
- Reduction of communication barriers between students and academics by offering different communication methods.

A concept of phases was designed to enrich the course with selected online offers even though the focus of the knowledge transfer continued on the phases of attendance. As early as in the conception period both limited available human resources and enormous pressure of time when implementing the course had to be assumed. To accommodate this aspect, an iterative proceeding was intended for the implementation of the online presence.

This proceeding is known from the software engineering. First, it implicates the advantages of a fast allocation of a first executable solution. Secondly, it provides potential to include student ideas (feedback) and step-by-step completion of the total solution. And thirdly, it gives a collection of practical knowledge in dealing with used tools. Beside these advantages, disadvantages of iterative proceedings also exist.

Particularly, these ones are diffuse technical specifications concerning the way contents of teaching can be conveyed with the help of an online platform. At any rate, the contents of the course and the required evidence of academic achievement should be exactly specified at the beginning of the lecture. In addition, the basic navigation within the teaching platform has to be set already at the beginning, too.

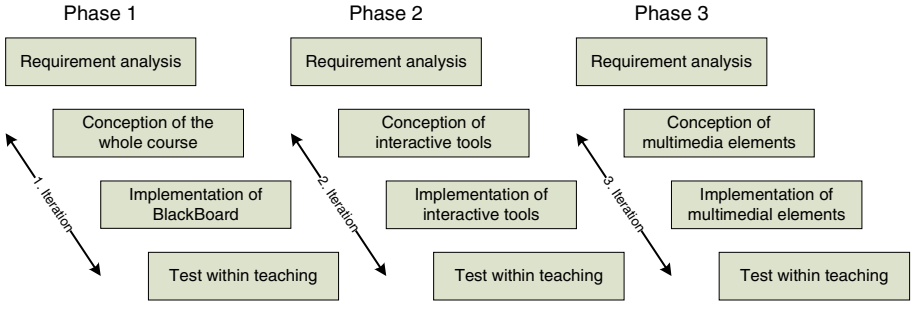


Fig. 1. Concept of phases within the course

According to the presented concept of phases (see figure 1), the course was divided into 3 phases which have to be passed through in an iterative way. During phase 1, only basic options of the used learning platform BlackBoard [BB08] were considered, whereas in phase 2 the utilization of the provided interactive tools of BlackBoard (chat and virtual classroom) occurred. In fact, the use of animated contents of teaching based on the usage of the Mediator-Tool was aimed for phase 3.

Right from the planning stage of the course, didactical, semantically supported planning mechanism like discribed in [SM07a] would have been helpful and would have contributed to a more efficient compilation.

3 Implementation

The implementation of the course reverted to already existing documents. An adjustment in form and content of the used materials to the opportunities of a Blended Learning course was especially carried out within the tutorial. The adjustment was especially necessary for the following aspects:

- Consideration of web-based possibilities
- Provision of prototypical implementations
- Hints for tools and techniques
- Hints for the project management
- Application of online questionnaires.

The implementation of the course within the BlackBoard platform proceeded extensively trouble-free. Only the formatting of some passages of the text was reverted to the possibility of direct editing within HTML-source code mode. The approach to the implementation was geared to the already depicted phases. According to this, the focus was on the particular functional demands. In no case, a randomly employment of as much as possible elements of the online platform should be effected.

Several technical problems were identified. Most relevant among them being the inadequately performance of the BlackBoard system, issues with handling of calendar functions (e.g. recurrent dates), an absent subscription mechanism (e.g. in case of modifications) as well as interoperability problems with the applied browser systems.

The conception assumed a more autonomous development of multimedia-based contents of teaching; whereas an already online available teaching platform was used in the context of the lecture. This platform was embedded into the BlackBoard presence. Reasons for this divergence of the concept can be primary seen in insufficient human resources and conceptual appendage of such a system. Furthermore, an analysis of already available solutions should be accomplished before the beginning of each implementation. It must be pointed out that an own implementation is not always a reasonable alternative solution. Automatic adaptation processes constitute an expedient improvement. They were exemplarily described amongst others within [SM08a].

4 Execution

4.1 Phase 1

One important aspect within the execution of a Blended Learning course refers to matters of initiation. Within the framework of the implementation required preconditions to an effective usage of online offers should be clarified. In detail, they deal with the following aspects:

- Allocation of required accounts and in case of Skype allocation of information about employed names. In the underlying course, accounts related to BlackBoard, Skype and the BPM platform.
- Review of technical requirements, such as efficiency of used computers, allocated network accesses and handling of inadequate resources.
- A joint discussion concerning each implicit goal of a Blended Learning course. In this process, not only expectations but also possibly existing anxieties of students should be detected.
- Another aspect of the implementation refers to a joint test of the provided online tools. In this context, the handling of optionally occuring problems (hotline) has to be clarified, too.

The employed educational material (lecture notes, tutorial documentation, exercises for project reports, further information) was allocated step-by-step in the progress of the course. This was carried out each 1 to 2 weeks before the actual lecture or tutorial. In addition, a document was provided on BlackBoard to specify the project report in form and content. Moreover, operational functions of polls, announcements and available mailing lists were used.

4.2 Phase 2

As already defined in the concept of the course, within phase 2 the usage of available interactive tools inside of BlackBoard (especially chat, virtual classroom and Skype) should be tested. In the framework of the first online session presentations of the achieved results from exercise 1 should be performed. Beyond, this online session should compensate for a temporal hesitation within the lecture. The preparation of this

session took place in line with two testing events at which all participants attended in one room. However, only feasibilities of the virtual classroom were tested, but not the utilization of Skype-accounts. It was not possible to avoid this restriction due to the students' usage of private computers during the online session. (see also appendix B1)

The virtual classroom was practised in connection with Skype. Genutzt wurde das virtuelle Klassenzimmer in Verbindung mit Skype. Even though the preparative interexchange of Skype names ran mostly without any difficulty (both participants have to be signed on indeed for a primary contact), Skype was not able to support meetings with more than 4 attendants within the actual session. While providing all functions of the virtual classroom without restrictions, the voice communication broke down. Furthermore, a bad voice quality was detected for the remaining participants (distortions, hall effects, durations).

According to the authors the virtual classroom which is currently available on BlackBoard is not suitable for cooperative conditioning of complex circumstances. In particular, the costs to invest for preparation of such a session are out of all proportion to the reachable result. In addition, the missing voice communication constricts a real interaction.

In the framework of further online sessions, only the Skype system was used due to the possibilities of an immediate message exchange as well as the chance to record contents of a session. Moreover, a frequent usage of Skype occured without a direct background of an intended online session. If Skype is not usable, an appropriate e-mail communication will adopt this task where necessary. However, deficiencies were diagnosed within Skype, too. But these deficiencies are not only caused by missing functionalities regarding e-Learning systems (amongst others to suggest a permanent availability).

In this context presented problems can be naturally avoided by dedicated communication platforms (e.g. already enabled through M-Bone at the beginning of the 1990s) or proactive adaptable systems (vgl. [SM07b]).

4.3 Phase 3

As mentioned before within the framework of implementation specification, phase 3 includes the integration of an online available teaching portal for animated illustration of the BPMN notation instead of the use of the mediator tool. With the assistance of such a system, at least one part of the intended content of teaching could be provided for the interactive private study within the BlackBoard-course. (see also Appendix B2)

Furthermore, interesting experiences could be gained on the base of this system. These experiences can influence the implementation of an own solution as well. In the following some aspects can be pointed out:

- An anglophone range of course offerings can settle claims in german academia only in a limited way. In this context, students mostly wish for contents presented in german-language.
- The used system only refers to the modelling of business processes. The changeover to the implementation (in a particular case the mapping on the XML-based languages BPEL and WSDL) is not regarded.
- The used system does not feature any options concerning an assessment of the training success. At this point, the integration of suitable queries or the attribution of imparted model elements respectively is desirable.
- The usage of an already available system does not hold the possibility to influence the didactical concept. In other words, the manner to procure contents can not be affected.

- In the same way, the availability and the performance of such a range of course offerings can not be ensured. At this point, it may be possible that it amounts to unsatisfying effects which have a direct impact on the reachable learning progress.

Appendix A presents the time schedule of the developed Blended Learning lecture with all 3 phases. It shows the lectures at the university in combination with online supported learning aspects.

5 Evaluation

The course was subject to an evaluation for several times within the semester. The evaluation consisted of a combination of immediate feedback during the lecture and polls which were implemented via BlackBoard system. In the following sections, selected results of the accomplished polls will be presented related to the attitude of expectation and the execution of the Blended Learning course. Furthermore, the cost and effort for the realized course will be compared to a classical approach. A complete demonstration was forgone for reasons of complexity. Further information can be gathered from the author.

5.1 Attitude of Expectation and Previous Knowledge

First, selected requested attitudes of expectation and as the case may be available previous knowledge of the students with handling of online platforms shall be elaborated. For acquisition of these data a poll with 10 questions in total was designed and provided within BlackBoard. Selected examples of questions and corresponding answers:

Questions A - Do you already have experiences with using online communities like the BlackBoard system which is employed in this case?

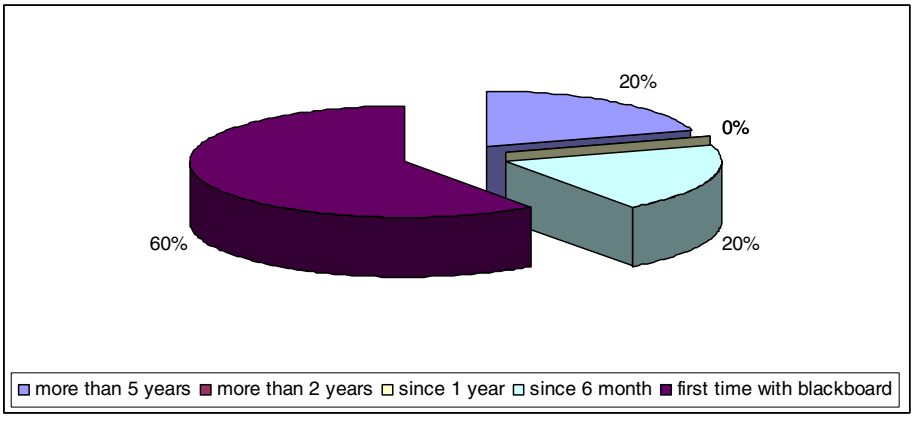


Fig. 2. Measuring of previous knowledge (Questions A)

Questions B - What do you think how long contents of the lecture and thereby worked out results shall be held out after the end of the course? Please give your answer in the unit of month!

In case of this short response, the wishes varied from “always” to pragmatical answers like “at least by the end of this semester”.

Questions C - How do you attach the importance of the usage of an online platform (as BlackBoard) to the execution of a course in academia?

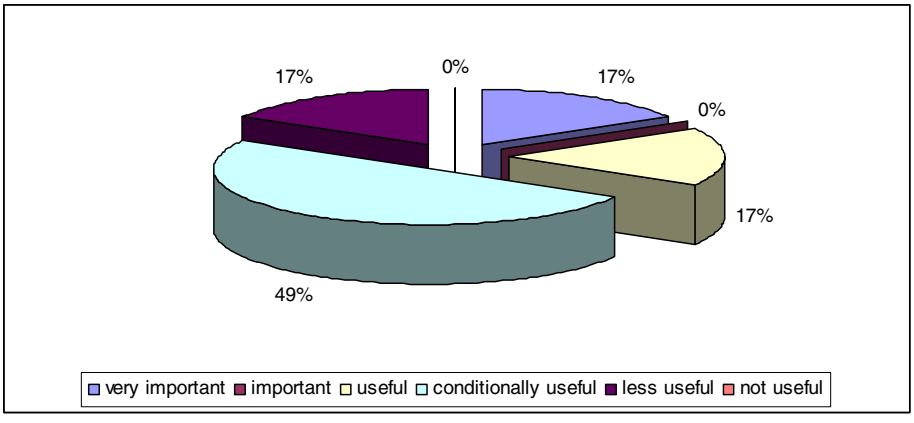


Fig. 3. Importance of an online platform in education (Questions C)

Questions D - How many in-class lectures should be replaced by corresponding online courses?

When answering this question at least 50 % of the respondents think that the substitution by corresponding online courses is not expedient. Nevertheless, 50 % decided in favour of a replacement of 10% by according online offers.

Note from the authors: Such a conservative attitude towards the usage of online offers as replacement of in-class lectures was not expected. The response astonishes, because the respondents were all students of the business informatics. It has to be checked if this attitude also applies to appending administrative systems (vgl. u.a. [SK07], [SM08b]).

Questions E - How fast should be reacted in the case of new contributions inside the discussion forum?

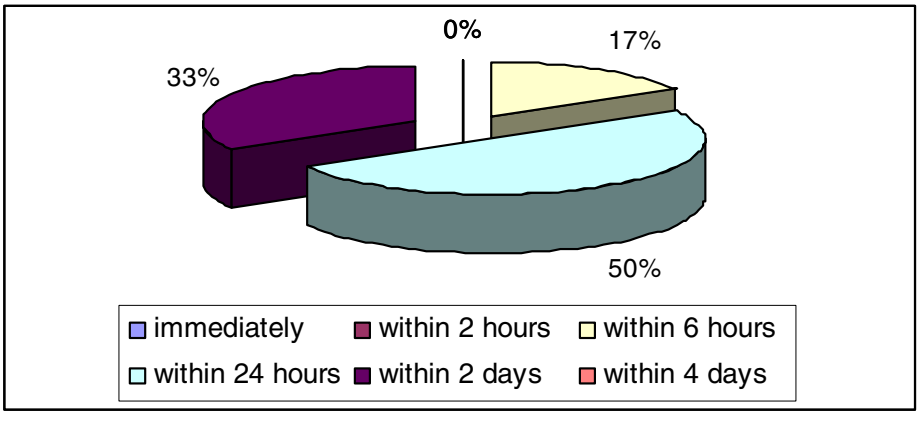


Fig. 4. Required reaction times (Questions E)

5.2 Feedback Concerning the Online Session

The received feedback on the online session refers to the use of the virtual classroom in combination to Skype within the second phase. A combination of discussions during in-class times and continued discussions using a forum was chosen for the feedback. Direct discussions regarding potential causes and the development of an approach to future meetings were essential after these obviously emerging problems. In addition, a pale disillusionment concerning the possibilities of a virtual session could be recognized for all participants. Therefore, it was necessary to continue active motivations.

The following statements of students (lightly modified in expression) give a good impression of constructive criticism. The structure of possible answers was given by the lecturer:

What I did like very well: The possibility to finish exercises and to present the results even though lectures were cancelled. To gain experiences with the handling of a new platform.

What I did not like: The shortcomings of the platform fully affected. Was already know and therefore possible to prepare for.

Are there any ideas for the improvement of the preparation: It might be reasonable to give a presentation without showing on the transparency, but only in a verbal way.

Reasonability of used tools: Skype: reduction on essential connections; Skype Messenger: was function used?; Chat: expedient to exchange short messages or links; Whiteboard: handling was not that easy.

Further additions to the tools: one microphone and camera per each group, marks on the transparencies for process stepping, Skype features a whiteboard, too, integrated function of conference call.

5.3 Comparison of Blended Learning and In-Class Lectures

Finally, surplus and deficit within the execution of this course have to be considered. Inside of the in-class lectures approximately 6 hours were spent on the explanation and the test respectively of the used BlackBoard tools. The following costs appeared on both sides *student* and *lecturer*:

- Introduction and test of basic BlackBoard tools: **1 hour**
- Repeated test of the possibilities of the virtual classroom: **3 hours**
- Adjustment of an approach to virtual meetings: **1 hour**
- Discussion about occurred problems: **1 hour**

The in-class times were not reduced for the execution of the virtual sessions. As a result, additional expenses arised out of it not only for the *students* but also for the *lecturers*:

- Preparation of the virtual meeting by e-mail or Skype-names: **1 hour**
- Execution of a virtual session with all participants: **2 hours**

- Execution of Skype-sessions with selected attendees: **2 hours**
- Response of the conducted polls: **0,5 hour** (*only students*)

To support students, an online presence already existed within prior semesters. According to this aspect, only certain operating expenses for administration of the BlackBoard presence can be accounted for additional investment concerning the *lecturer*. These include:

- Implementation of the structure of the BlackBoard course: **2 hours**
- Implementation und administration of the placed fora: **1 hour**
- Implementation and evaluation of polls: **6 hours**
- To synchronize the course with the online possibilities: **10 hours**

To sum up, the extra work load for the execution of the course accounted for **24 hours** (at 60 semester hours) for the lecturer. These hours can exclusively be associated with the use of the possibilities of the online course. The reduction of the in-class time by **6 hours** has to be considered as well since it is not available for the work on the lecture topics any longer. Not included are the times which were spent on editing the courseware in form and context (inclusively animated presentations) because these ones emerge independent from the used online platform. However, the allocation of materials as well as the communication with all students could be simplified.

6 Conclusions for Software Engineering

Based on experience from the described course we are able to derive many possibilities to support Software Engineering activities. An e-teaching course during the early phase of a software development project can help to improve the learning curve. Over it out project members are able to use online possibilities step by step. Also an established project can reach benefits from a blended-learning approach. In the following one, selected examples are shown:

- Experience collection under use of online functionalities
- Support of a distributed software development
- Support during problem situations (online-consultant)
- Support of the agile software development paradigm
  - Pair programming through the use of online-tools (e.g. e-whiteboard)
  - Feedback possibilities (e.g. forum based or questionnaires)
  - Refactoring (e.g. online source code reviews)
  - “Online”-site customer (e.g. use of messaging components)
- Support of quality assurance activities (e.g. web-based prototypes)
- Use of animation and simulation possibilities of a new solution or process.

In any case the implementation of a blended learning course requires resources. The introduction of tools is not enough. Important is the concept behind the tools. The

experiences from "blended learning" courses at the universities could drive the implementation of "blended consulting" approaches.

7 Summary and Outlook

Within this paper, the conception, the compilation and the implementation of a Blended Learning course was described in detail. In addition, used potentials and advantages in the field of modern teaching concepts were specified as well as occured imponderabilities. In conclusion, the concept justifies its existence even though it has to be supported with appropriate methods and tools. The manual course compilation requires high costs.

Especially in the framework of the general development of Mediator-tools the need for further research is given. For instance, questions regarding the software-technical design and the approach to the documentation of such a system developed just as the efforts for the implementation which have to be taken into account. Within the scope of a current accomplishing research project the following facts have to be clarified amongst others: Which contents should be imparted with the aid of a multimedia-based teaching unit? How does reasonable module segmentation can be carried out? How do adequate use cases look like? How do assessments of training success can be represented in an effective way? Which aspects might a Style Guide have to consider? The Implementation of a simple prototype already required approximately 6 hours. Therefore, it is necessary to conduct realistic effort estimations for an appropriate allocation of human resources concerning the implementation of such a project.

References

[AS08] American Society for Training and Development: e-Learning Glossary (29.02.2008) (2008), http://www.learningcircuits.org/glossary

[BB08] Blackboard: Learning Management System (29.02.2008) (2008), http://www.blackboard.com

[SK07] Kernchen, S., Kunz, M., Dumke, R.R.: Proactive Class Schedule. IEEE Multidisciplinary Engineering Education Magazine 2(3), 24–28 (2007)

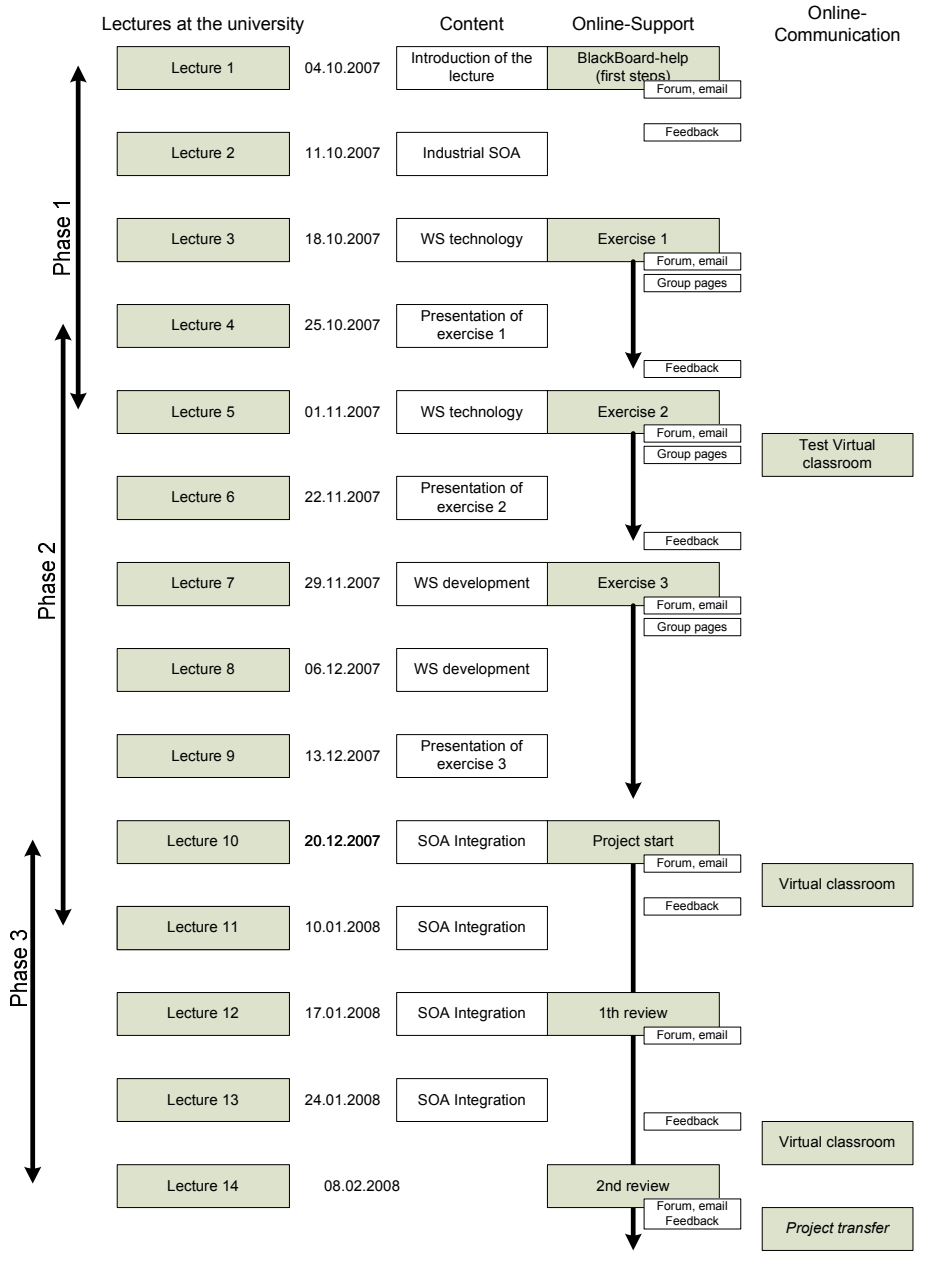
[SM07a] Mencke, S., Dumke, R.R.: A Hierarchy of Ontologies for Didactics-Enhanced E-learning. In: Proceedings of the International Conference on Interactive Computer Aided Learning (ICL 2007), Villach, Österreich, September 26-28 (2007)

[SM07b] Mencke, S., Dumke, R.R.: Developing Adaptive and Self-Managed Graphical User Interfaces. In: Proceedings of the International Conference on Interaction Mobile and Computer Aided Learning (IMCL 2007), Amman, Jordanien, April 18-20 (2007)

[SM08a] Mencke, S., Rud, D., Zbrog, F., Dumke, R.R.: Proactive Autonomous Resource Enrichment For E-Learning. In: Proceedings of the 4th International Conference on Web Information Systems and Technologies (WEBIST 2008), Funchal, Madeira, Portugal, pp. 4–7. Mai (2008)

[SM08b] Mencke, S., Kunz, M., Dumke, R.R.: Steps to an Empirical Analysis of the Proactive Class Schedule. In: Proceedings of the International Conference on Interaction Mobile and Computer Aided Learning (IMCL 2008), Amman, Jordanien, April 16-28 (2007)

Appendix A – Time Schedule of the Lecture



Appendix B –Time Schedule of the Lecture

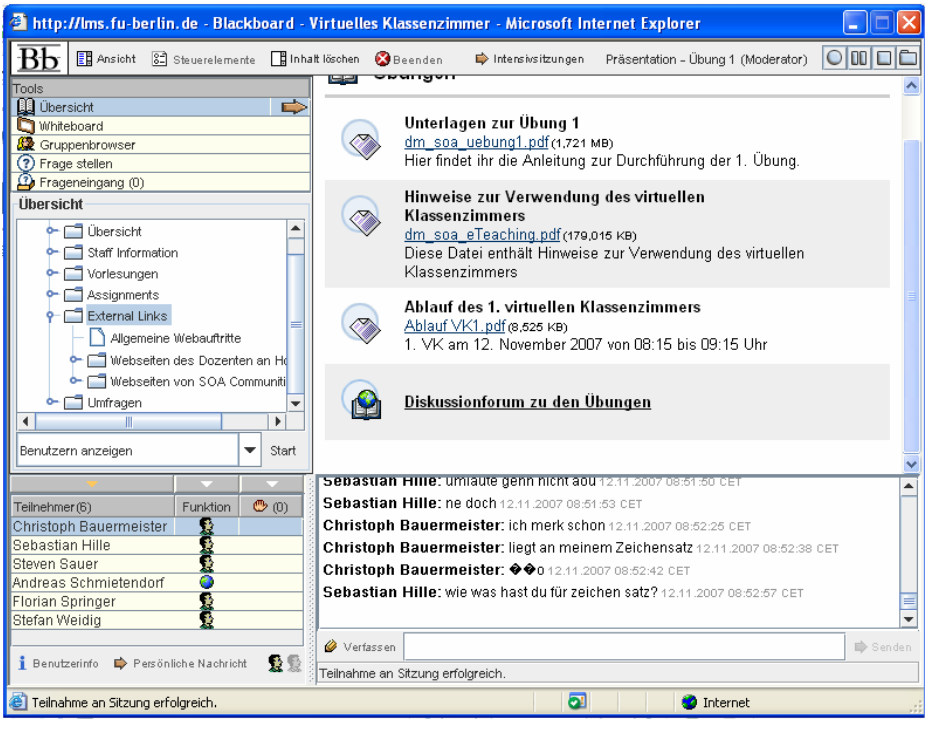


Fig. B1. Use of the virtual classroom

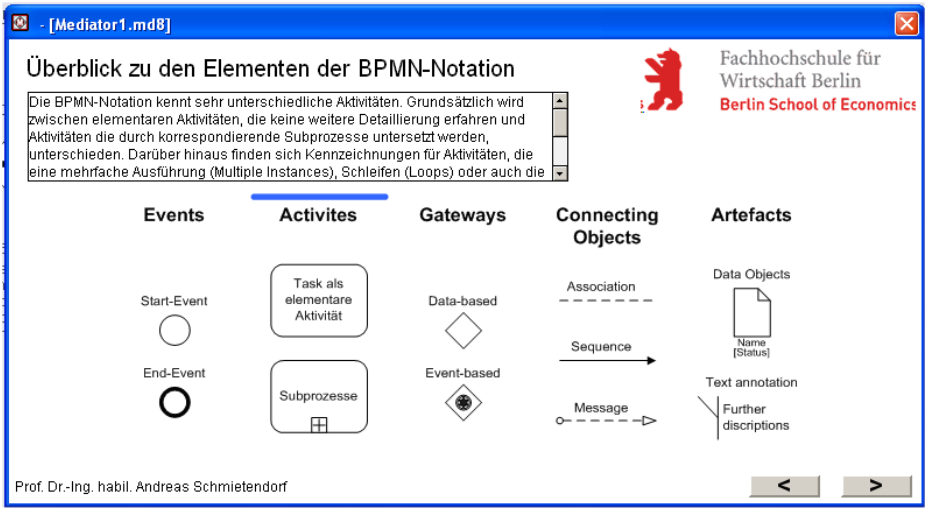


Fig. B2. Multimedia-supported teaching unit

How to Use COSMIC Functional Size in Effort Estimation Models?

Cigdem Gencel

Blekinge Institute of Technology - Department of Systems and Software Engineering, Ronneby, Sweden
Cigdem.Gencel@bth.se

Abstract. Although Functional Size Measurement (FSM) methods have become widely used by the software organizations, the functional size based effort estimation still needs further investigation. Most of the studies on effort estimation consider total functional size of the software as the primary input to estimation models and they mostly focus on identifying the project parameters which might have a significant effect on the size-effort relationship. This study brings suggestions on how to use COSMIC functional size as an input for effort estimation models and explores whether the productivity values for developing different functionality types deviate significantly from a total average productivity value computed from total functional size and effort figures. The results obtained after conducting a multiple case study in which COSMIC method was used for size measurement are discussed as well.

Keywords: Functional Size Measurement, Effort Estimation, Functionality, COSMIC, Base Functional Component.

1 Introduction

Since the first introduction of Function Point Analysis (FPA) method by Albrecht in 1979 [5], Functional Size Measurement (FSM) methods have not only been improved, but also new variations and extensions have been developed to be able to measure the new types of applications [18][43].

Among these methods, the ones which conform to ISO/IEC 14143-1 standard [22][23] are accepted as international standards for FSM. Common Software Measurement International Consortium Full Function Points (COSMIC-FFP) [26][14], International Function Point Users Group (IFPUG) FPA [27][20], MarkII FPA [28][45], Netherlands Software Metrics Association (NESMA) FSM [29][41] and Finnish Software Metrics Association (FiSMA) FSM methods are the ones accepted as FSM standards up to now [30].

Besides the usage of software size for a number of reasons such as in project tracking or for normalization of other measures, one of the major uses of software size is that it is the primary input for most effort and cost estimation models.

However, effort estimation based on functional size still remains a challenge for software practitioners and researchers. As more empirical data are collected in benchmarking

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 196–207, 2008.

datasets, the studies to explore the nature of the relationship between functional size and effort has been arisen. Taking the functional size as the main input, most of the studies on effort estimation investigate some project related attributes which might have impact on functional size and effort relationship. Unfortunately, the common conclusion of the existing studies is that although different models are successfully used by different groups and for particular domains, they do not have unanimous acceptance by the software community they being not performing well enough.

Traditionally, the functional size of a software system is measured as a single total value obtained by a specific FSM method. All FSM methods have their own attribute definition models [37], which derive this single size value by expressing a relationship among the sub-attributes, called Base Functional Component (BFC) Types.

In [37], Kitchenham et al claimed that Function Points (FP) might be viewed as a means of measuring the "shape" of a software product in terms of a vector of significant elements. They added that if the elements of such a shape measure influence the product development effort, an attempt could be made to derive an effort estimation model based on these elements.

In other engineering disciplines, there are different representations of the size for the same product. For example, in civil engineering, different size measures are defined to size buildings [13][12]. A vector of measures such as the floor area -which is calculated by multiplying the length and width of the floor- and the height of the building, is one representation. Or, a derived measure such as the volume of a building which is calculated by the multiplication of length, width and height of the building is another. The selection depends on the needs of the engineers or managers. For example, if the volume measure is sufficient for effort and cost estimation purposes in a specific project, this measure is used. Similarly, we also need different representations of software size for different purposes in software engineering.

In our previous studies [17][11], we investigated whether effort estimation models based on Base Functional Component (BFC) types[1], rather than those based on a single total functional size value would improve estimation reliability. The assumptions of these studies were that the amount of effort to be utilized for developing a unit size of different BFC types would be different. For the empirical study, we used the projects data in the International Software Benchmarking Standards Group (ISBSG) dataset [21] and formed sub-sets based on different criteria. We made the statistical analysis on the projects which were measured by COSMIC method. We made multiple regression analysis for investigating the strength of the relationship between the functional sizes of BFC Types and development effort. The results of both studies showed significant improvement in the size-effort relationship.

In this study, we propose a new representation of COSMIC functional size to be used in effort estimation models. Instead of a single size figure, we define a vector of measures. We identify the elements (or sub-attributes) of functional size which provide different functionalities to the users considering the BFC Types and the effort collection mechanisms in software organizations. We also present the results of a multiple case study which we conducted to explore whether the productivity values

[1] BFC: "an elementary unit of FUR defined by and used by an FSM Method for measurement purposes". BFC Type: "a defined category of BFCs. A BFC is classified as one and only one BFC Type" [23].

for developing each element deviates significantly from a total average productivity value computed from total functional size and effort figures.

2 Background

2.1 Functional Size Measurement

Albrecht's 1979 proposal [5] for estimating the functional size became a significant contender for software size measurement and hence effort estimation. This method was aimed at overcoming some of the shortcomings of measures based on Source Lines of Code (SLOC) for estimation purposes and productivity analysis, such as their availability only fairly late in the development process and their technology dependence. The FPA method is based on the idea of measuring the amount of functionality delivered to users in terms of Function Points (FP) taking into account only those elements in the application layer that are logically 'visible' to the user and not the technology used. FPA was designed in a Management Information System (MIS) environment and has become a *de facto* standard in the MIS community.

During the following years, variations of the original method have been developed[2]. Some of them either provided unique viewpoints different from the dominant method of their time or extended the applicability of FSM methods to different functional domains in addition to business application software such as Real-time systems, Web applications, etc. Other methods designed to measure software which are developed using object oriented methodology.

In the '90s, work was initiated at the ISO level to lay the foundations for regulating standards in FSM, and the 14143 family [24][25][31][32][33] was developed with five instantiations matching with those requirements: COSMIC-FFP [26][14], IFPUG FPA [20][27], MkII FPA [45][28], NESMA FSM [41][29] and FiSMA FSM [30] methods.

Albrecht's original idea has become the basis for IFPUG FPA [3], one of the earliest ISO standardized FSM methods [20][27]. IFPUG FPA enjoys widespread popularity and large publicly available data sets for those who wish to train their own company-specific IFPUG model or to compare their measurements with others.

MkII FPA [44] was developed by Symons in 1988 in order to improve the original FPA method. This method brought some suggestions to reflect the internal complexity of a system. Currently, the Metrics Practices Committee (MPC) of the UK Software Metrics Association (UKSMA) is the design authority of the method [45]. It was also mainly designed to measure business information systems. Mk II FPA has been accepted as being conformant to ISO/IEC 14143 and become an international ISO standard in 2002 [28].

NESMA FPA [41] has the same rules as the IFPUG FPA method. The differences between these two methods is due to NESMA measurement manual provides different guidelines, hints and examples. It was accepted by ISO as an international standard in 2005 [29].

COSMIC-FFP [14], adopted in 2003 as ISO 19761 [26], has been defined as a 2nd generation FSM method as a result of a series of innovations, such as: a better fit with

[2] Please refer to [18] and [43] for a detailed discussion on and a history of FPA-like methods.

both real-time and MIS environments, identification and measurement of multiple software layers, different viewpoints from which the software can be observed and measured, and the absence of a weighting system.

Finally, FiSMA FSM, accepted as an international FSM standard in 2008 [30], was developed by a working group of FiSMA. It is a general parameterized size measurement method that is designed to be applied to all types of software. The difference of FiSMA FSM from other methods is that it service-oriented rather than process-oriented.

2.2 Software Size Based Effort Estimation

In parallel to these developments in FSM, significant research has been going on software size based effort estimation such as in [7][8][19][34][35][36]. In [6][16][40][42], significant variations in the impact of project cost drivers have been observed. Among the cost drivers investigated, Team Size, Programming Language Type, Organization Type, Business Area Type, Application Type, Development Type and Development Platform have been found to affect the size-effort relationship at different levels of significance.

In a number of studies such as [2][9][10][39], the related works on estimation models are assessed and compared. However, the common conclusion of these studies was that although different models are successfully used by different groups and for particular domains, none of them has gained general acceptance by the software community.

Other studies such as [1][3][17][11] focused merely on functional size and explored different ways to use it as an input to effort estimation models. In [1][3], the concept of software functional profile is defined as the relative distribution of its four BFC Types for any particular project. They investigated whether or not the size-effort relationship was stronger if a project was close to the average functional profile of the sample studied. It was observed that the identification of the functional profile of a project and its comparison with the profiles of their own samples can help in selecting the best estimation models relevant to its own functional profile.

In [17][11], it was explored whether effort estimation models based on the BFC types rather than those based on a single total value would improve estimation models. Both of these studies showed significant improvement in modeling the size-effort relationship.

The results of the literature survey show that functional size based effort estimation still require further investigation. This paper focuses on investigating how to use functional size in effort estimation models. A new representation of COSMIC functional size, which can be used for effort estimation purposes, will be defined.

3 Suggestions for a New Representation of COSMIC Functional Size

In COSMIC measurement process [15], the Functional User Requirements (FURs) are decomposed into their elementary components, called “Functional Processes”. A Functional Process is defined as “an elementary component of a set of FUR comprising a unique, cohesive and independently executable set of data movements”.

A Data Movement Type is defined as "a BFC which moves one or more data attribute types belonging to a single data group type". There are four kinds of Data Movement Types: Entry, Exit, Read, and Write. Each of these is defined as a BFC Type in [15] as;

- An *Entry* is a data movement type that moves a data group from a functional user across the boundary into the functional process where it is required.
- An *Exit* is a data movement type that moves a data group from a functional process across the boundary to the functional user that requires it.
- A *Read* is a data movement type that moves a data group from persistent storage within reach of the functional process which requires it.
- A *Write* is a data movement type that moves a data group lying inside a functional process to persistent storage.

After identifying the BFC Types in the Functional Processes, the second step involves calculating the functional size of each BFC by applying a measurement function to the BFC Types and the related attributes. Then the results are aggregated to compute the overall size of the software system.

Each of these BFC Types represents different types of functionalities to be provided to the users. That is the reason why FSM methods identify these elements to measure functional size. In our previous studies [17][11], we investigated whether effort estimation models based on the BFC types rather than those based on a single total value would improve estimation models. Both of these studies showed significant improvement in modeling the size-effort relationship. This approach has the potential to result in generic effort estimation models when significant amount of projects data can be collected in benchmarking datasets. However, until then, software organizations might collect their own data so that average productivity figures to develop each type of functionality can be found at least for that specific organization.

Since we are after finding a representation of functional size for effort estimation purposes, we also have to consider the effort collection mechanisms in software organizations in addition to the BFC Types which serve different functionalities. Based on these, we identify the significant elements for which we define a vector of measures.

In software development projects, if the components of a software product are developed by using different technologies, implemented on different processors or developed by utilizing different programming languages, the efforts utilized for each are can usually be identified. The components involve development of different types of functionalities. At a first glance, we identify two possible functionality types related to interface and data services components.

The Entry and Exit data movement types are the functionalities provided to the functional user for moving data groups across the boundary. We call them Interface functionalities.

The Read and Write data movement types are the functionalities provided to the functional user for moving data groups from persistent storage within reach of the functional process and to the persistent storage lying inside a functional process. We call them Data Services functionalities.

In the first version of COSMIC method, data type characteristics in business applications and in real-time systems were investigated [38][4]. The differences between the single-occurrence control data in real-time systems and the multiple-occurrence

group of data in business application software are discussed. Although the units of measure in COSMIC [15] are the same for measuring these different functionalities and the functional sizes of different BFC Types can be added to compute the total size, the amount of effort to be utilized per unit size might be different. Therefore, we also differentiate Business-Application Data Services from Control Data Services.

Thus, the new representation of COSMIC Functional size for effort estimation purposes involves a vector of measures for the following elements: Interface, Business-Application Data Services and Control Data Services. This new representation does not interfere with any of the principles of COSMIC. This is analogous to using the same unit of measure and measurement rules for measuring the length, width, and height of a building. The difference is that we do not use a compound measure as the volume measure, but a vector of measures for representing COSMIC functional size.

4 Case Study

We conducted a multiple-case study in order to evaluate the proposed representation for COSMIC functional size. Our research question for this case study was the following: "Are the productivity figures for developing each element, i.e. each functionality type, deviates significantly from the total average productivity figure for developing the whole software?"

We designed this case study as a multiple-case study which involves three new development case projects. We applied COSMIC to measure the functional size of the case projects. The case projects and the case study are described and discussed in the following sub-sections.

4.1 Description of the Case Projects and Organizations

Project-1 involves the development of a military inventory management system integrated with a document management system. The software development organization is an independent supplier, which is a CMMI/SW Maturity Level 3 company. The organization focuses mostly on web-based projects and has its own framework to develop web applications rapidly. The project was started in October 2004 and completed in August 2005. 7 persons worked for the project: 1 project manager, 1 senior software engineer, 2 software engineers (development team – full-time), 2 software engineers (development team - part-time), and 1 software engineer (test team – part-time).

Project-2 involves the development of a multimedia sponsored call system. The system enables advertising companies to relay their messages to their target market through call sponsorships and enables end-users to have sponsorships for their calls by receiving an interactive multimedia advertisement at the beginning or during the call. The software company develops telecommunications solutions and provides network infrastructure components for the telecommunications industry, having totally 80 personnel 50 of which are engineers. The company owns TSE-EN-ISO 9001:2000 quality certificate. 9 persons worked for the project: 1 project manager, 1 senior software engineer (development team leader), 6 software engineers (development team) and 1 software test engineer (test team leader).

Project-3 involves the development of an equipment identification registrar which detects and warns the operator against potential fraud risks such as Subscriber Identity

Module (SIM) card cloning and International Mobile Equipment Identity (IMEI) cloning. The same software company which developed Project-4 developed this project as well. 10 persons worked for the project: 1 project manager, 2 senior software engineer (development team leader), 6 software engineers (development team) and 1 software test engineer (test team leader).

The total efforts utilized for the software life cycle processes of the case projects are 6,308, 1,080 and 1,200 person-hours for Project-1, Project-2 and Project-3, respectively.

We used the CHAR Method [33] to determine Functional Domains of the case projects. The functional domains of Project-1, Project-2 and Project-3 are 'Information System', 'Complex Controlling Information System' and 'Information System, respectively.

Table 1. Functional Domains of the Case Projects determined by CHAR Method [33]

Project No	Functional Domain	Control- and communication-rich FUR	Data-rich FURs	Manipulation- and algorithm-rich FURs
1	Information System	Negligible	dominant	present
2	Complex Controlling Information System	Present	dominant	present
3	Information System	Negligible	dominant	present

4.2 Case Study Conduct and Data Collection

All the projects were measured by COSMIC FFP v.2.2 [15] utilizing the Software Requirements Specification (SRS) documents prepared according to the companies' SRS standards.

Two measurers together measured the functional size of Project-1 (see Table 2). One of the measurers works for the development organization and is involved in this project. The other is the author of this paper. Both are experienced in using COSMIC. The effort utilized to make measurement is 13 person-hours.

One measurer performed the functional size measurement for the other two projects (see Table 2). She is the author of this paper. The efforts utilized to make measurement are 15 person-hours for each of the other two projects.

The functional sizes of Project-1, Project-2 and Project-3 were measured as 1020, 321 and 275 COSMIC Function Points (CFP), respectively.

The research question in this case study was to explore whether the elements of functional size influence the product development effort, i.e. whether the Productivity Delivery Rates (PDR) to develop different elements might be different.

Table 2. Case Projects COSMIC FFP size measurement details

Project No	Number of Functional Processes	Number of Entries	Number of Exits	Number of Reads	Number of Writes	Functional Size (CFP)
Project-1	127	154	378	333	155	1,020
Project-2	50	80	79	99	63	321
Project-3	54	69	115	45	46	275

The PDR values for the projects, which are calculated as the ratio of effort to total COSMIC functional size, are given in Table 3. Since all of these projects involve algorithmic operations which cannot be measured by COSMIC method, we excluded these efforts from the total effort values.

For the case projects, the effort values were collected in detail so that the amount of effort utilized for each functionality type could also be identified. For Project-1, the software coding and unit testing efforts were collected based on the three types of functionalities. Therefore, for this project we calculated the PDR value based on Code and Unit Test Effort values. In this project, the Interface and the Permanent Storage/access functionalities were developed by using the Internal Development Framework (IDF) which was developed by the development organization. IDF is a tool to reuse CRUDL processes in standard web applications. By this tool, the interface and database components are generated in parallel with each other. For the processing component, Java is used as the primary programming language. These components were developed not only by different teams but also using different technologies.

For Project-2 and Project-3, the effort data are available for the whole development. Therefore, we calculated the PDR values for developing these different functionality types. In Project-2, Java was the primary programming language and in Project-3, Java and ANSI were the primary programming languages.

Table 3. PDR Values of the Projects for the Elements of COSMIC Functional Size

		Total Figures	The Elements of COSMIC Functional Size		
			Business-Application Data Services	Control Data Services	Interface
Project-1	Functional Size (CFP)	1,020	488	0	532
	Code &Unit Test Effort (person-hours)	1.333	742	-	591
	PDR (person-hrs/CFP)	*1.31*	*1.52*	-	*1.11*
Project-2	Functional Size (CFP)	321	146	16	159
	Development Effort (person-hours)	1,010	540	200	270
	PDR (person-hrs/CFP)	*3.15*	*3.70*	*12.50*	*1.70*
Project-3	Functional Size (CFP)	275	91	0	184
	Development Effort (person-hours)	1,130	450	-	680
	PDR (person-hrs/CFP)	*4.11*	*4.95*	-	*3.70*

4.3 Discussion of the Case Study Results

Since Project-2 and Project-3 were developed by the same organization, we investigated the deviation between PDR values of these two projects. The PDR of Project-3 deviates 30.5% from Project-2. The PDR values for developing each of the key size parameters are shown in Table 4.

Table 4. % Deviation in PDR for the elements from the Average PDR

Projects	***% Deviation in PDR for the COSMIC elements from the Average PDR***		
	Business-Application Data Services	Control Data Services	Interface
Prj-1	16.03	-	-15.26
Prj-2	17.46	296,82	-46.03
Prj-3	20.44	-	-9,97

The results show that, for all the projects the amount of effort required to develop business-application data services per unit size is greater than the average figures whereas it is less for developing the interface functionalities. For Project-2, PDR value for developing Control data services is so high. Therefore, its deviation from the average figure is also very high.

5 Conclusion

This study aimed to investigate whether a different representation of COSMIC functional size without changing any rules and principles of the method, would have the potential to improve effort estimation reliability.

Different functionality types are identified by grouping COSMIC BFC Types considering the effort collection mechanisms in the software organizations. Accordingly, Interface, Business Application Data Services and Control Business Data Services are identified as being different functionality types.

The case study results showed that there is a significant variation between the PDR values for developing different kinds of functionalities. Therefore, building estimation models using this new representation for COSMIC functional size rather than using a single total value is promising.

Moreover, by this representation of size, we get more information about the functional domain of the software and the types of functionalities to be provided to the users.

Acknowledgements

I would like to thank Pinar Efe and Figan Bilgin for their contributions in performing the case studies.

References

[1] Abran, A., Gil, B., Lefebvre, E.: Estimation Models Based on Functional Profiles. In: International Workshop on Software Measurement – IWSM/MetriKon, Kronisburg, Germany, pp. 195–211. Shaker Verlag (2004)

[2] Abran, A., Ndiaye, I., Bourque, P.: Contribution of Software Size in Effort Estimation, Research Lab. In: Software Engineering, École de Technologie Supérieure, Canada (2003)

[3] Abran, A., Panteliuc, A.: Estimation Models Based on Functional Profiles. III Taller Internacional de Calidad en Technologias de Information et de Communications, Cuba, (February 15-16, 2007)

[4] Abran, A., St-Pierre, D., Maya, M., Desharnais, J.M.: Full Function Points for Embedded and Real-Time Software. In: UKSMA Fall Conference, London, UK, October, pp. 30–31 (1998)

[5] Albrecht, A.J.: Measuring Application Development Productivity. In: Proc. Joint SHARE/GUIDE/IBM Application Development Symposium, pp. 83–92 (1979)

[6] Angelis, L., Stamelos, I., Morisio, M.: Building a Cost Estimation Model Based on Categorical Data. In: 7th IEEE Int. Software Metrics Symposium (METRICS 2001), London (April 2001)

[7] Boehm, B.W.: Software Engineering Economics, p. 487. Prentice-Hall, Englewood Cliffs (1981)

[8] Boehm, B.W., Horowitz, E., Madachy, R., Reifer, D., Bradford, K.C., Steece, B., Brown, A.W., Chulani, S., Abts, C.: Software Cost Estimation with COCOMO II. Prentice Hall, New Jersey (2000)

[9] Briand, L.C., El Emam, K., Maxwell, K., Surmann, D., Wieczorek, I.: An Assessment and Comparison of Common Software Cost Estimation Models. In: Proc. of the 21st Intern. Conference on Software Engineering, ICSE 1999, Los Angeles, USA, pp. 313–322 (1999)

[10] Briand, L.C., Langley, T., Wieczorek, I.: A Replicated Assessment and Comparison of Software Cost Modeling Techniques. In: Proc. of the 22nd Intern. Conf. on Software Engineering, ICSE 2000, Limerick, Ireland, pp. 377–386 (2000)

[11] Buglione, L., Gencel, C.: Impact of Base Functional Component Types on Software Functional Size based Effort Estimation. In: Jedlitschka, A., Salo, O. (eds.) PROFES 2008. LNCS, vol. 5089, pp. 75–89. Springer, Heidelberg (2008)

[12] CESMM - Civil Engineering Standard Method of Measurement, 3rd edn. Thomas Telford Ltd. (1991)

[13] Chen, W.F., Liew, J.Y.R.: The Civil Engineering Handbook, 2nd edn. CRC Press, Boca Raton (2003)

[14] COSMIC. COSMIC- v.3.0, Measurement Manual (September 2007)

[15] COSMIC: The Common Software Measurement International Consortium FFP, version 3.0, Measurement Manual (2007)

[16] Forselius, P.: Benchmarking Software-Development Productivity. IEEE Software 17(1), 80–88 (2000)

[17] Gencel, C., Buglione, L.: Do Different Functionality Types Affect the Relationship between Software Functional Size and Effort? In: Cuadrado-Gallego, J.J., et al. (eds.) IWSM-Mensura 2007. LNCS, vol. 4895, pp. 72–85. Springer, Heidelberg (2008)

[18] Gencel, C., Demirors, O.: Functional Size Measurement Revisited. ACM Transactions on Software Engineering and Methodology (TOSEM) 17(3), 71–106 (2008)

[19] Hastings, T.E., Sajeev, A.S.M.: A Vector-Based Approach to Software Size Measurement and Effort Estimation. IEEE Transactions on Software Engineering 27(4), 337–350 (2001)
[20] IFPUG. Function Points Counting Practices Manual (release 4.2), International Function Point Users Group, Westerville, Ohio (January 2004)
[21] ISBSG Dataset 10 (2007), http://www.isbsg.org
[22] ISO/IEC 14143-1: Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts (1998)
[23] ISO/IEC 14143-1: Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts (February 2007)
[24] ISO/IEC 14143-2: Information Technology – Software Measurement – Functional Size Measurement - Part 2: Conformity Evaluation of Software Size Measurement Methods to ISO/IEC 14143-1:1998 (2002)
[25] ISO/IEC 14143-6: Guide for Use of ISO/IEC 14143 and related International Standards (2006)
[26] ISO/IEC 19761:2003, Software Engineering – COSMIC-FFP: A Functional Size Measurement Method, International Organization for Standardization (2003)
[27] ISO/IEC 20926:2003, Software Engineering-IFPUG 4.1 Unadjusted Functional Size Measurement Method - Counting Practices Manual, International Organization for Standardization (2003)
[28] ISO/IEC 20968:2002, Software Engineering – MK II Function Point Analysis – Counting Practices Manual, International Organization for Standardization (2002)
[29] ISO/IEC 24570:2005, Software Engineering – NESMA functional size measurement method version 2.1 – Definitions and counting guidelines for the application of Function Point Analysis, International Organization for Standardization (2005)
[30] ISO/IEC 29881:2008, Software Engineering – FiSMA functional size measurement method version 1.1, International Organization for Standardization (2008)
[31] ISO/IEC TR 14143-3: Information Technology – Software Measurement – Functional Size Measurement – Part 3: Verification of Functional Size Measurement Methods (2003)
[32] ISO/IEC TR 14143-4: Information Technology – Software Measurement – Functional Size Measurement - Part 4: Reference Model (2002)
[33] ISO/IEC TR 14143-5: Information Technology – Software Measurement – Functional Size Measurement – Part 5: Determination of Functional Domains for Use with Functional Size Measurement (2004)
[34] Jeffery, R., Ruhe, M., Wieczorek, I.A.: Comparative Study of Two Software Development Cost Modeling Techniques using Multi-organizational and Company-specific Data. Information and Software Technology 42, 1009–1016 (2000)
[35] Jorgensen, M., Molokken-Ostvold, K.: Reasons for Software Effort Estimation Error: Impact of Respondent Role, Information Collection Approach, and Data Analysis Method. IEEE Transactions on Software Engineering 30(12), 993–1007 (2004)
[36] Kitchenham, B., Mendes, E.: Software Productivity Measurement Using Multiple Size Measures. IEEE Transactions on Software Engineering 30(12), 1023–1035 (2004)
[37] Kitchenham, B., Pfleeger, S.L., Fenton, N.: Toward a Framework for Software Measurement Validation. IEEE Transactions on Software Engineering 21(12) (December 1995)
[38] Maya, M., Abran, A., Oligny, S., St-Pierre, D., Desharnais, J.M.: Measuring the Functional Size of Real-Time Software. In: Proc. of 1998 European Software Control and Metrics Conference, Maastricht, The Netherlands, pp. 191–199 (1998)
[39] Menzies, T., Chen, Z., Hihn, J., Lum, K.: Selecting Best Practices for Effort Estimation. IEEE Transactions on Software Engineering 32(11), 883–895 (2006)

[40] Morasca, S., Russo, G.: An Empirical Study of Software Productivity. In: Proc. of the 25th Intern. Computer Software and Applications Conf. on Invigorating Software Development, pp. 317–322 (2001)
[41] NESMA. Definitions and Counting Guidelines for the Application of Function Point Analysis, v.2.0 (1997)
[42] Premraj, R., Shepperd, M.J., Kitchenham, B., Forselius, P.: An Empirical Analysis of Software Productivity over Time. In: 11th IEEE International Symposium on Software Metrics (Metrics 2005), p. 37. IEEE Computer Society, Los Alamitos (2005)
[43] Symons, C.: Come Back Function Point Analysis (Modernized) – All is Forgiven!). In: Proc. of the 4th European Conference on Software Measurement and ICT Control, FESMA-DASMA 2001, Germany, pp. 413–426 (2001)
[44] Symons, C.: Function Point Analysis: Difficulties and Improvements. IEEE Transactions on Software Engineering 14(1), 2–11 (1988)
[45] UKSMA. MkII Function Point Analysis Counting Practices Manual, v 1.3.1 (1998)

Uncertainty in ERP Effort Estimation: A Challenge or an Asset?

Maya Daneva[1], Seanna Wettflower[2], and Sonia de Boer[2]

[1] University of Twente, The Netherlands
[2] ProMetrix, Toronto, Canada
m.daneva@utwente.nl, {wettflower,deboer}@prometrix.ca

Abstract. Traditionally, software measurement literature considers the uncertainty of cost drivers in project estimation as a challenge and treats it as such. This paper develops the position that uncertainty can be seen as an asset. It draws on results of a case study in which we replicated an approach to balancing uncertainties of project context characteristics in requirements-based effort estimation for ERP implementations.

1 Introduction

Adopting ERP solution means, more often than not, considerable investments of time, money, and effort. Therefore, for a company, the decision to roll-out an ERP package has major implications. An ERP implementation program may well take years, particularly if the ERP adopter changes both the package and the organization itself in order to achieve a better alignment between its business processes and the system components which support them [1,3,24,28]. Moreover, the huge effort and difficulties associated with ERP roll-outs caused the ERP endeavors a notorious reputation: market research firms who study the success and failure rates of ERP projects indicate that at least 85% of the ERP implementation projects are delayed or are over budget, 60% end up with reduced scope, and 35% are cancelled [3]. This situation lets ERP adopters perceive uncertainties of project context as a huge challenge as it's almost impossible for ERP-adopters to determine a level of trust in any estimate. Examples of some specific barriers to trust, identified in published research [1,12,16,19, 24,26,27,28] include: lack of consensus on the objectives of the estimates, no known steps to ensure the integrity of the estimation process, no historical evidence at the ERP adopter's site supporting a reliable estimate, or the inability to clearly see whether or not estimates are consistent with consultants' demonstrated accomplishments on other projects in comparable organizations in the same sector.

Traditionally, the software measurement community confronts a problem situation like this one by proposing effort estimation solutions [10] with the intent to help companies (i) get an increased understanding of the impact of their specific context on project scope, effort and delivery dates, (ii) reason about the resources to be consumed in a project, and (iii) become aware of how uncertainties of project context matter. In the ERP area, however, so far very few remedies [6,27] have been proposed to the above challenges. This paper contributes to the discussion on ERP effort

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 208–222, 2008.

estimation and the role of context uncertainties therein. We present the application of one particular empirical approach to balancing uncertainties, by investigating the relationship of uncertain cost drivers on project success under time and under effort constraints. Our findings suggest that uncertainties should not necessarily be viewed as challenges in ERP projects. Based on data we collected, analyzed, and reflected upon, we revise the 'uncertainties-as-challenges' position and consider uncertainties as an asset. Assuming that uncertainties are an asset, an implication for ERP adopters would be, than, that their ERP project managers or program directors may want to consider some project context characteristics for adjustment, so that the chance of their projects' success are increased.

In the next section we provide empirical background on our research questions. In section 3, we sketch our empirical approach and in Section 4, we provide a step-by-step description of its application in the case study site. In Section 5, we address some validity threats and in Section 6, we draw our conclusions.

2 Empirical Background, Related Work, and Research Questions

ERP implementation has spawned an independent industry with firms providing consulting services exclusively and generating total revenue of billions of dollars. Because it has existed for less than 15 years, the business practices of requesting proposals for ERP implementation services, of bid preparation, and of pricing are all haphazard. ERP adopters find themselves, increasingly more often than ever, in a scenario where they need to compare bidding information from competing implementation service providers and initiate negotiations with them. For ERP adopters to be adequately prepared for this exercise, they need to acquire knowledge (i) on how each consulting firm arrived at the bidding price and (ii) on how realistic the effort estimation figures which one sees on the bidding document, are. However, at that early stage of requirements engineering (RE), uncertainties of context interfere greatly with adopter's ability to assess to what extent the price they receive from the bidding document matches their organizational realities. Today, it's well known that a typical ERP project includes diverse configurations, each of which matches the needs of a unique stakeholder group, which, in turn, implies the presence of cost drivers unique to each configuration. Moreover, at time of bid preparation (that is, the stage of very early requirements), consulting firms would have a relatively low level of awareness of what new project activities (e.g. identifying and analyzing capability gaps, investigation and mapping of configuration options [22]) are to be added in order to plan and manage the ERP project, and what the ERP adopter's context factors are that drive effort for these new activities. Yet, ERP adopters must find a way to balance uncertainties of their contexts at that very early stage. This issue (of how to treat uncertainties) has been approached by management scientists and by software measurement practitioners and both communities have come up with some solutions.

Quantitative studies by management science scholars have shown how ERP adopters can use financial valuation techniques when it comes to evaluating investments in large ERP assets under uncertainty. These studies have presented the mechanics of Real Option Analysis (RO) and portfolio management models [2,29] and have clearly demonstrated the benefits of these two techniques. However, when

studying the industrial intake of these techniques, we found that their practical application was still very limited [9]. For ERP adopters, using these techniques remains a challenge, because of two reasons: (i) these techniques are very data-intensive and (ii) they heavily rely on the presence of financial valuation experts specialized in the application of these techniques to IT (or ERP) assets [29].

In the past five years, the software measurement community proposed solutions to uncertainties be incorporated into traditional effort estimation techniques (e.g. COCOMO II [4]) by using concepts of fuzzy logic or of probability theory [13]. For example, instead of using 'data points' as inputs into algorithmic models of effort estimation, one could and should consider representing uncertain inputs by using probability distributions. These uncertain inputs are, then, processed by means of some simulation techniques, for example a Monte Carlo simulation [13,21] or a Latin Hypercube simulation [15]. For estimation analysts, a recognized [13] advantage of using such an approach is the ability to calculate the implications of uncertainties (by means of the simulations). For a project manager, being aware of these implications, the critical pragmatic issue, is, then, this: *which project context characteristic s/he can change, so that his/her actions would cause a desired change in an important project outcome*. We make the note that in the case of ERP implementation projects, such a probability-theory-based approach to uncertainties has been tried out relatively recently (in 2007). However, the above-mentioned advantage has been experienced in the case study [6], in which the approach was used. The approach itself complementarily applied a traditional effort estimation technique – COCOMO II [4], with two uncertainty-handling mechanisms, namely, Monte Carlo simulation [21] and a probabilistic portfolio management model [11]. In its application it was found that for certain context characteristics among the ones captured by COCOMO, it is possible to adjust their ratings in a way which maximizes the project success. This motivated us to look into similar but different settings to replicate the use of our approach and increase our understanding of how uncertainties can be balanced so that success is maximized.

Based on the discussion in this section, we formulate the following three research questions we are set out to answer in this paper:

RQ1: Does the adjustment of uncertain cost drivers in the COCOMO II model increase the chance of success?

RQ2: Which cost drivers of the COCOMO II model can be adjusted in a way that maximized the chance of success under time constraints?

RQ3: Which cost drivers of the COCOMO II model can be adjusted in a way that maximized the chance of success under effort constraints?

We make the note that answering RQ1 brings insights into RQ2 and RQ3. This implies that when a project manager recognizes those context factors that improve the success chances of a portfolio of projects, s/he can steer the projects rationally even under considerable troubled condition. In the next sections, we propose pragmatic aids for getting these questions answered in a systematic way. We defined a research method that is constructive, based on the literature survey [9], complemented with our own practical experience with mid-sized and large ERP projects in the telecommunication sector and in the sector of financial services.

3 The Empirical Research Method

The present study is aimed at identifying and understanding uncertainties in ERP effort estimation at the ERP bidding stage, at which point requirements are not yet fully known. That means (i) we follow [18] in that we consider a project quote to consist of three components, estimated cost, profit, and contingency, and (ii) we focus on the models used to estimate cost in particular. We do not cover issues pertaining to profit and contingency. We aim to use the findings from this exploratory study as a basis for more in-depth studies in this area.

We do our case study research by using the initial model of investigating uncertainties proposed in [6,7]. The model is a first attempt in the direction of balancing ERP project context uncertainties and we expect it to be refined over a period of time. The study described here is a replication of the study reported in [6,7] and carried out in two business units of a company in the telecommunication sector with two ERP systems (SAP and PeopleSoft, in each business unit, respectively) implemented by three Canadian ERP service providers. This earlier case study completed by the first author had two key objectives: to demonstrate the practical advantages of the solution approach and to carry out a proof-of-concept [32]. In the present case study, we replicate the use of the approach in order to collect evidence which would possibly strengthen our claims which were drawn from the findings in [6]. For practitioners to be able to draw more general conclusions and to (dis)confirm the results obtained in our first case study [6], we now replicated the study using data collected from sites in a large company in a different service sector (namely financial service provisioning) where implementation was facilitated by two US consulting firms. The shared context characteristic between the first case study [6] and this one is that both organizations implemented the same ERP package, namely SAP.

In carrying out this study, we have followed the recommendations of Yin [34] for case study research. In what follows we first provide a summary of the three techniques that compose our initial model to investigate uncertainties and then, we explain its application in the case study settings.

3.1 Description of the Three Techniques

This case study applies the following three techniques:

(i) the COCOMO II reference model [4] which we used to account for ERP adopter's specific cost drivers,
(ii) the Monte Carlo simulation [21] which lets us approach the cost drivers' degrees of uncertainty, and
(iii) the effort-and-deadline-probability-based portfolio management concept [11] which lets us quantify the chance for success with proposed interdependent deadlines for a set of related ERP projects.

Below we summarize the key ideas in each technique and how they all fit together. The overall design of the approach is elaborated in more detail in [6]. In Section 4, we demonstrate its application in the present case study.

COCOMO II [4]: This is one of the best-known algorithmic model for setting budgets and schedules as a basis for planning and control. It comprises (i) five scale factors, which reflect economies and diseconomies of scale observable in projects of various sizes, and (ii) 17 cost drivers, which serve to adjust initial effort estimations. In ERP project settings, at least three of the scale factors are directly related to the joint RE and architecture design activities, and thus raises the role of architects in reducing project costs. COCOMO II allows ERP teams to include in their estimates (i) the maturity level of the ERP adopting organization, (+ii) the extent to which requirements' and system architecture's volatility is reduced before ERP configuration, and (iii) the level of team cohesion and stakeholders' participation. In COCOMO II, the degrees of both the scale factors and the cost drivers vary from extra low, very low, low and nominal to high, very high and extra high. Suppose ERP project stakeholders assign a degree to each scale factor and cost driver, the estimation of project effort and duration will result from the two equations below:

$$\text{Effort} = A \times (\text{Size})^{E} \times \prod_{i=1}^{17} EM_i \quad (1)$$

$$\text{and} \quad \text{Time} = C \times (\text{Effort})^{F} \quad (2)$$

where E and F are calculated via the following two expressions, respectively:

$$E = B + 0.01 \times \sum_{j=1}^{5} SF_j \text{ and}$$

$$F = D + 0.2 \times (E - B)$$

In (1) and (2), *SF* stands for the scale factors, and *EM* means cost drivers.

Monte Carlo simulations: This is a problem-solving technique used to approximate the probability of certain outcomes by running multiple trial runs, called simulations, using random variables. We used it here to counterpart the inherent uncertainty of the cost drivers by applying NOSTROMO [21], a Monte Carlo simulation technique used at the THAAD Project Office (USA). When used in combination with COCOMO II, repeatedly running the model many times and collecting samples of the output variables for each run helps the estimation analysts produce an overall picture of the combined effect of different input variables distribution on the output of the model.

Portfolio management [11]: This is an effort-and-deadline-probability model that allows us to quantify the uncertainty associated with a project estimate. Its merits are that (i) it is applicable at the stage of requirements or project bidding [11], (ii) its only input requirement is a record of previous projects; and (iii) it fits with the ERP adopters' project realities suggesting that an ERP project is implemented as a portfolio of interdependent subprojects [5,6]. Each subproject is a piece of functionality (or an ERP module) linked to other pieces (or modules). For example, the Sales and Operations Planning component in a package is tightly linked with the Cost Center Planning functionality of the Controlling module. Suppose we have a set of interdependent subprojects, the effort-and-deadline-probability model [11] will yield (i) the probability of portfolio's success with the proposed deadlines for each subproject in this

portfolio, and (ii) a set of new deadlines which will result in a required probability of success. The portfolio success is judged by two conditions applied to any two subprojects ***a*** and ***b*** for which deadline$_a$ is earlier than deadline$_b$. The conditions are that: (i) subproject ***a*** is to be over by deadline$_a$ and (ii) subproject ***a*** and subproject ***b*** are to be over by deadline$_b$. In other words, the conditions require all subprojects planned with a deadline before deadline$_b$ to be completed by deadline$_b$, rather than just project ***b***. This is the key to the portfolio approach, because uncertainty about completion of project ***b*** incorporated uncertainty from all previous projects.

Suppose the ERP adopter engages in total E people in the project and let d be the number of work days it takes from start date to deadline, then the total available resources is Exd. So, suppose an ERP portfolio $\boldsymbol{Y}$ is made up by n subprojects, the success conditions are represented as follows:

$$\begin{pmatrix} Y_1 \\ Y_1 + Y_2 \\ \cdots \\ Y_1 + Y_2 + \ldots Y_n \end{pmatrix} \leq E \begin{pmatrix} d_1 \\ d_2 \\ \cdots \\ d_n \end{pmatrix} \tag{3}$$

where Y_i is the estimated effort for subproject i to succeed. We check if, for any j, (j= 1..n), the sum of $Y_1,..,Y_j$ is greater of Exd_j. If this is true, then deadline d_j has failed. Success probabilities result from simulations in which $Y_1,...,Y_n$ are generated from a predetermined probability distribution. If we deem $Y_1, \ldots, Y_n$ is satisfying all conditions, then we say that the portfolio $\boldsymbol{Y}$ succeeds. The portfolio's probability of success is equal to the ratio of the number of successes in the set $\boldsymbol{Y}$ to the number of trials in the simulation.

3.2 Making the Three Techniques Work Together

The eight steps of applying our approach are presented in Figure 1. The procedures pertinent to each step in Figure 1 have been designed with the initial intention to serve the case study purposes. However, practitioners can use them to compare and evaluate the roles of cost drivers in increasing the chances of success of their projects, and also make decisions on which drivers' ratings should be adjusted. We make the note, that because our approach is meant for the RE project stage, we consider **Unadjusted Function Points** (FP) [8] to be used as a size estimate. This is consistent with the position of the COCOMO II authors [4, see p. 17]. Moreover, the first author has devised a technique for applying it at the stage of early requirements for ERP projects [8].We chose this measure of functional size because (i) it is applicable to any ERP package and not to a specific package's context and (ii) it's the only measure of size that fits the project stage of early requirements. Furthermore, to account for uncertainty of the ERP project context, we suggest the COCOMO II model take as inputs the probability distributions of the five COCOMO scale factors and 17 cost drivers, instead of using as inputs single values (as in [4]). This design choice has been recommended by the THAAD Project Office [21] and by the CAIG researchers [15] as well. Deploying the Monte Carlo simulation means to ascribe a particular distribution type to an input variable in a model, get randomly-selected values, feed them into the

COCOMO II model and, then, see how likely each resulting outcome is. In other words, for each uncertain factor, our approach yields possible effort and duration estimation values. In contrast to COCOMO II, our output is the probability distributions of effort and duration and not the most likely effort and duration (which COCOMO II creates).

The probability distributions are, then, fed into the portfolio management method [11]. To run it, we first formulate a condition for success, as in (3), then we bunch projects into portfolios and we obtain the probability of successfully delivering the projects under time constraints as well under effort constraints.

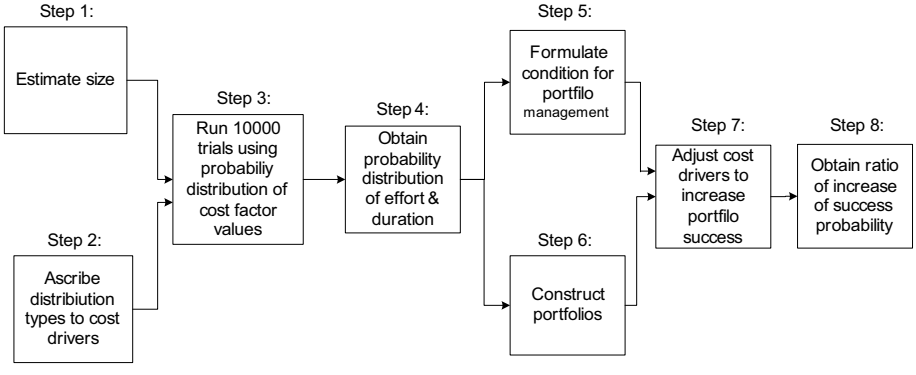


Fig. 1. The step-by-step application of the approach: a high-level view

4 The Replication Case Study

This section provides a description of the context in which the steps in Figure 1 were executed, the results obtained and the conclusions we derive from this empirical work effort.

4.1 Context of the Application of the Method

The solution approach was applied in a setting of a large multi-site ERP roll-out that included six functional modules of one ERP package (namely SAP). These modules were: Material Management, Sales and Distribution, Human Resources, Accounts Payable, Accounts Receivable, and Asset Management. Our data came from 19 SAP projects implemented in the case study company between March, 2001 and June, 2005. In this period, the second author was an internal SAP consultant responsible for the implementation of two modules. The ERP implementations relied on the involvement of consultants of two companies: one was a major SAP implementation services provider in Nord America and the other was a smaller consulting company involved as a subcontractor. The implementation process model adopted in the context of the projects was the proprietary SAP implementation model of the major SAP consulting company. To describe the practical settings of the projects we follow the schema for describing project contexts used by the first author in her earlier ERP studies [6]. We decided to do so, because this would provide a common ground for project comparison (should we decide

to compare lessons learnt from our past case studies and recent ones), for carrying out meta-analysis in the future and for strengthening the external validity of case study findings [34]. The practical settings for our 19 projects included the following:

- to manage implementation complexity, each of our projects was broken down in a number of subprojects reflecting the number of functional components to be configured. For example, the first project had to implement three components and was broken down in three subprojects. The total number of our subprojects in which the consulting company's implementation process was instantiated was 71.
- for each subproject, there was a dedicated RE team. This is a group of individuals who are assigned to a specific subproject, contribute time to and run the RE cycle for this subproject, and deliver the business requirements document for a specific SAP component. Each RE team consisted of one or two SAP consultants who provided in-depth knowledge in both the implementation process and the SAP components, and one or two numbers of business representatives, the so-called process owners. They were department managers and/or subject matter experts who contributed the necessary line know-how, designed new processes and operational procedures to be supported by the SAP modules, and provided the project with the appropriate authority and resources.
- all process owners had more than 15 years of experience with IT-projects in their departments. We deem this 'above average' level. Before starting the projects, attended a full-day training workshop on the SAP implementation methodology designed by the consulting company.
- the consultants formed a mix of experts (65%) and new hires (35%). Each expert had at least 8 years of configuration and integration experience with a specific SAP functional module. Most experts had spent many years in the consulting company and knew the implementation process in much detail. With respect to the new hires, the two consulting companies provided evidence that their less experienced staff-members have been SAP-certified specialists for the modules they were supposed to work on. Half of the consultants had solid experience in the financial services sector, and half of them did not. The latter were unaware of the requirements principles in this domain and were supposed to carry out RE activities under novel conditions. All the teams were supported by five process architects responsible for architecting the solution, sharing process knowledge and consulting on ongoing basis with the teams on SAP reuse, process methods, and RE tools. Each architect was a resource which was shared by 3-4 RE teams. The 71 teams worked separately and with relatively little communication among them. This allowed us to initially consider and include 71 subprojects in our case study.

In what follows, we describe how the steps from Figure 1 have been applied:

Step 1: For each of the 19 projects, we got (i) project size data, (ii) start and end dates, and (iii) scale factor and cost driver ratings. Size was measured in terms of unadjusted IFPUG FP according to the counting rules for SAP projects presented in [8]. The effort multipliers *A, B,* and *EM* in equation (1) and (2) and the scale factors *SF* were calibrated by using ERP effort data collected between 2001 and 2005 in the case study company.

Step 2: The case study found that the company provided the ratings of the cost drivers and scale factors only and there was contradicting knowledge about the uncertainty of these ratings. For this reason, we assigned to each factor its distribution type

and its parameters of probability distribution (namely center, minimum and maximum) based on previously published experiences and recommendations by other authors [17,21]. For example, in this case study, we used McDonnald's [21] default 'high' levels of uncertainty associated to the ratings of the RESL, DATA, ACAP and PCAP cost drivers [4]. (Because of space limitation, we refer readers to reference [4] which gives detailed definitions of these cost drivers). The level of uncertainty determines - in turn, the distribution type to be assigned to each cost driver:

- the normal distribution type is assigned to a low-uncertainty cost driver,
- the triangular distribution type is assigned to a medium-uncertainty cost driver, and
- uniform distribution type is assigned to a high uncertainty cost driver.

We make the note that we used a lognormal distribution for functional size, which was motivated by the observations of Chulani et al [5]. These researchers investigated the size distribution and indicate that its skew is positive and that log(size) is likely to be a normal distribution.

Step 3: Taking the COCOMO II factors and uncertainty values as input data, we run Monte Carlo simulations which produced two types of samples: (i) samples of effort, expressed in person-months, and (ii) samples of time, expressed in months. A typical Monte Carlo simulation consists of many - often thousands of, trials, each of which is an experiment where we supply numerical values for input variables, evaluate the model to compute numerical values for outcomes of interest, and collect these values for later analysis. In this case study, we used 10000 trials.

Step 4: Running the trials generated the samples of effort and time, as presented in Figure 2 and Figure 3, respectively. In these histograms (also known as 'density charts [15]'), the Y-dimension (on the right of the figure) shows the frequency with which a value was observed in the sample of 10000 trials. The X-dimension shows the value range. Because the average subproject involved four professionals (two business users, one external consultant and one architect) we adopted the assumption for E to be 4.

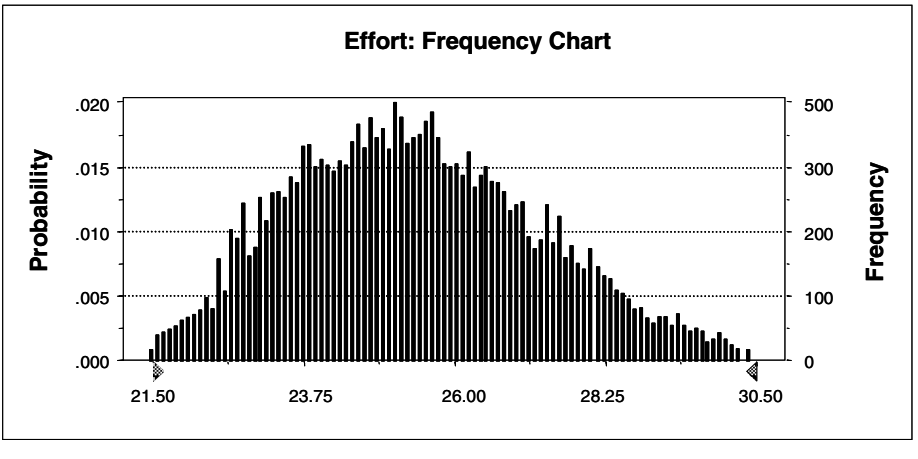


Fig. 2. The Monte Carlo histogram of the probability distribution of effort (in person/months)

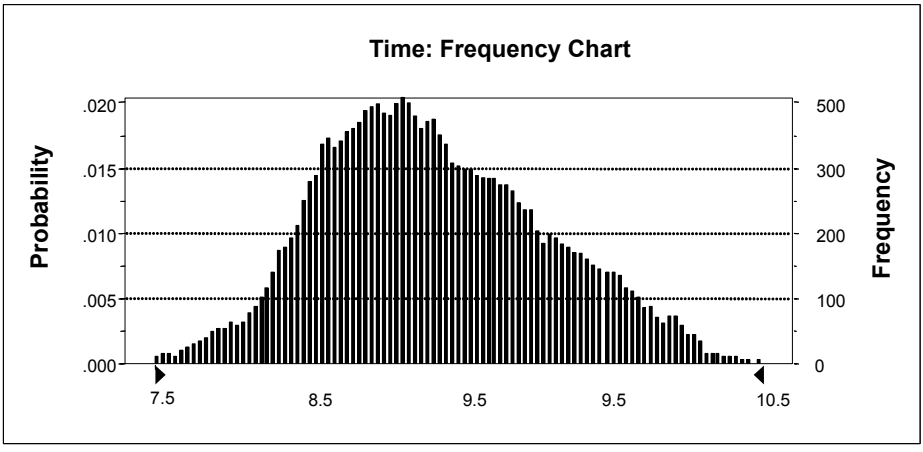


Fig. 3. The Monte Carlo histogram of the probability distribution of time (in months)

Step 5: Drawing on the observation that COCOMO II provides time estimation as in (**), we formulated the following condition for portfolio management in terms of time constraints:

$$\begin{pmatrix} T_1 \\ T_1 + T_2 \\ \dots \\ T_1 + T_2 + \dots T_n \end{pmatrix} \leq \begin{pmatrix} m_1 \\ m_2 \\ \dots \\ m_n \end{pmatrix} \tag{4}$$

where T_i is the ERP implementation time in months for subproject i. In this condition, we did not include the number of people E, because COCOMO II assumed an average number of project staff [2] which was accounted in (2).

Step 6: We attempted to improve the chances for portfolio success by adjusting the cost drivers and scale factors. Hence, we adopted the assumption that *for projects with two different ratings for the same cost driver or scale factor, the probability of success for each project will be different too.* To understand how cost drivers and scale factors make a difference in terms of project success, for each cost driver/scale factor we constructed two portfolios: the first one had this driver/cost factor rated 'very high' for all projects and the second portfolio had it rated 'very low' for all projects. This point is illustrated in more detail in the next section. (For example, we found that when selective reuse [6] was practiced in ERP projects, the probability of success was higher under both time and effort constraints.)

Steps 7–8: We run the portfolio management method [11] to obtain a probability of success under time constrains and under effort constraints.

4.2 Results

This sections reports on the results with respect to what we observe when adjusting COCOMO II cost drivers. Because of space limitation, we could not provide the data we obtained when adjusting each of the 17 COCOMO II cost drivers and the 5 scale factors. Here, we report on the results (see Table 1) we obtained when adjusting the driver termed SITE [4].

Table 1. Analysis of the probability of success for the factor SITE under effort constraints and time constraints

SITE rating	Probability of success	
	Under effort constraints	Under time constraints
Very high	66.91%	82.52%
Very low	83.11%	97.88%

As specified in Step 6 (Fig. 1), we constructed two portfolios of subprojects, namely the first one with the factor of SITE rated as *very high* for all subprojects and the second one with SITE rated very low for all subprojects. We make two notes: First, that *low* level of SITE indicates a centralized-across-business-units data processing concept and that a high level of SITE indicates business-unit-specific (or site-specific) customization of both data flows and control flows [5]. Second, we ruled out the rating 'extremely low' as it's relatively rarely to be observed in a ERP project context [20,23,24]. Table 1 suggests that when a project is composed of subprojects all of which have SITE rated *very low*, the probability of success is greater under both time and effort constraints.

We observed that 13 out of the 17 factors from the COCOMO II model can be adjusted in a way that maximizes the probability of success. These 13 factors are: database size (DATA), product complexity (CPLX), REUSE, documentation (DOCU), platform volatility (PVOL), analyst capability (ACAP), programmer capability (PCAP), personnel continuity (PCON), applications experience (APEX), language and tool experience (LTEX), use of software tools (TOOL), multi-site implementation (SITE), required implementation schedule (SCED).

5 Discussion on Threats to Validity

This is the first replication study of the approach originally described in [6] and carried out in a similar but different setting. Clearly, at this stage of research, though replicated, we deem our results preliminary. Threats to validity were considered, based on definitions of Wohlin et al [33]: First, the major threat to external validity arises from the fact that the company's projects might not be representative for the entire population of ERP adopters. Because the case study company was ranked by ERP market research firms among the top-ten-percent most successful ERP adopters, there is a possibility that the company is more mature in ERP implementation than the average financial services firm in Nord America. That this might be the case can be concluded also by the matter that (i) the company had on board a large number of

experienced staff members and (ii) consultants with expert knowledge of SAP and required SAP-certification proofs for those new hires of the consulting companies who were on the project team. It's interesting to see if results would be different if we use as a case study site a less mature financial services firm. However, we make the note that in terms of implemented modules, we allows us to consider the case study company's project context typical for the financial services companies in North America: we judge these settings typical because they seemed common for all SAP adopting organizations who were members of the American SAP Financial Services User Group (ASUG). The ASUG meets on regular basis to discuss project issues and suggest service sector-specific functionality features to the vendor for inclusion in future releases. The SAP components our case company implemented are the ones which other ASUG companies have in place to automate their non-core processes (accounting, inventory, sales & distribution, human resources).

Second, when constructing the portfolio, each author based her choice of 'very low/ very high' ratings on her own experience in implementing ERP. We noticed that the experiences of the two authors, who were employed in the past in SAP consulting roles, varied with respect to five cost drivers. This might be a result of the different backgrounds of these authors (the first – in the telecommunication sector, and the second – in the financial industry). In any case, the authors set up the ratings in a way that - clearly, is subjective. As done in [6], we make the note that this design choice was the only possible way to go, given the fact that, to the best of our knowledge, there is no published research on the COCOMO II factor ratings which are more common in ERP context. We plan, in the future, to research the topic of economies and diseconomies of scale in ERP projects, hoping that new knowledge will help refine our approach.

Next, we deployed complementary three models of three types and are well aware of other possibly useful techniques by each type. Instead of COCOMO II, one may decide to use COSYSMO [30]. Instead of Monte Carlo simulations, the Latin Hypercube technique could be used [15]. Instead of the portfolio management model by Fewster and Mendes [11], the portfolio approach of Chris Verhoef [31] might be good candidates for inclusion. In the future, we are interested in investigating whether different modeling choices sustain our results or limit the validity of our findings to the subset of the analyzed models. We make the note, however, that when it comes down to the practical applicability of the various combinations of techniques by these three types, it all will depend on the availability of data by certain types at ERP adopter's sites. So, the choice of techniques should represent an acceptable balance between data-intensiveness and usefulness. As pointed in Section 2, some techniques are more data-intensive than others and, therefore, we expect that their usefulness will vary. We expect that those techniques which are more data-intensive will be of more limited use, as very few ERP organizations have the practice of disciplined project data collection and reporting.

6 Conclusions

This replication study has demonstrated that it's possible to adjust cost drivers so that project managers increase the probability of success for highly uncertain ERP

projects, a company might have to implement. This brought us to the idea that we can view uncertainty in early ERP projects as an asset and a resource to project managers. We have also found that 13 out of the 17 COCOMO II cost drivers can be adjusted to increase the chances for success. This result converges with a finding in an earlier case study by the first author [6]. This result also agrees with the understanding among the software practitioners [14] that when organizations reflect on their discrepancies between the estimated and the actual effort cost, this reflection might lead to improving the adopters' assessments of uncertainty[1] .

We acknowledge the possible validity threats [33] as our most important issue and in the next year, we will work in collaboration with three European companies to carry out a series of experiments and case studies. The results will serve to properly collect evidence supporting or weakening our claims and evaluate their validity. This will serve our ultimate objective to get an improved version of our method.

Acknowledgements

We thank all parties involved in this study. The first author also thanks the following organizations without whose support this research program would not have become a reality: the Netherlands Science Organization (NWO) for supporting the CARES project and the QuadREAD project, and the CTIT organization for supporting the COSMOS project.

References

[1] Arnesen, S., Thompson, J.: How to Budget for Enterprise Software, Strategic Finance, pp. 43–47 (January 2005)
[2] Bardhan, I., Bafgci, S., Sougstad, R.: A Real Options Approach for Prioritization of a Portfolio of Information Technology Projects: a Case Study of a Utility Company. In: Proc. of the 37th Hawaii Int'l Conf. on Systems Sciences (2004)
[3] Benchmarking Partners, A Status Report on ERP Implementation, USA (2007)
[4] Boehm, B.: Software Cost Estimation with COCOMO II. Prentice Hall, Upper Saddle River (2000)
[5] Chulani, S., Boehm, B., Steece, B.: Bayesian Analysis of Empirical Software Engineering Cost Models. IEEE Trans on SE 25(4), 573–583
[6] Daneva, M.: Managing Uncertainty in ERP Effort Estimation Practice: an Industrial Case Study. In: Int'l Conference on Software Process and Product Improvement, Frascati. LNCS. Springer, Heidelberg (2008)
[7] Daneva: Approaching the ERP Project Cost Estimation Problem: an Experiment. In: Int'. Symposium on Empirical Software Engineering and Measurement (ESEM), p. 500. IEEE Computer Society Press, Los Alamitos (2007)
[8] Daneva, M.: Measuring Reuse of SAP Requirements: a Model-based Approach. In: Proc. of Symposium on Software Reuse. ACM Press, NY (1999)

[1] While this hypothesis was first proposed and investigated by Gruschke and Jorgensen [14] in a non-ERP software project context, we do not see why it would not apply to ERP settings.

[9] Daneva, M., Wieringa, R.J.: Cost Estimation for Cross-organizational ERP Projects: Research Perspectives. Journal of Software Quality (2008)
[10] Fenton, N.E., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach, 2nd edn. PWS Publishing (1998)
[11] Fewster, R.M., Mendes, E.: Portfolio Management Method for Deadline Planning. In: Proc. of METRICS 2003, pp. 325–336. IEEE, Los Alamitos (2003)
[12] Francalanci, C.: Predicting the implementation effort of ERP projects: empirical evidence on SAP R/3. Journal of Information Technology 16, 33–48 (2003)
[13] Garvey, P.R.: Probability Methods for Cost Uncertainty Analysis—A Systems Engineering Perspective. Marcel Dekker Publ., New York (2000)
[14] Gruschke, T.M., Jørgensen, M.: Assessing Uncertainty of Software Development Effort Estimates: Learning from Outcome Feedback. In: ACM Transactions on Software Engineering and Methodology (accepted, 2008)
[15] Gupta, S., et al.: Considerations in Cost Risk Analysis: How the IC CAIG Handles Risk. In: SCEA 2002 National Conference, June 14 (2002)
[16] Hansen, T.: Multidimensional Effort Prediction for ERP System Implementation. In: Meersman, R., Tari, Z., Herrero, P. (eds.) OTM 2006 Workshops. LNCS, vol. 4278, pp. 1402–1408. Springer, Heidelberg (2006)
[17] Jiamthubthugsin, W., Sutivong, D.: Protfolio Management of Software Development Projects Using COCOMO II. In: Proc. of ICSE 2006, pp. 889–892 (2006)
[18] Kitchenham, B.A., Pickard, L., Linkman, S., Jones, P.: Modelling Software Bidding Risks. IEEE Transactions on Software Engineering 29(6), 542–554 (2003)
[19] Koch, S.: ERP Implementation Effort Estimation Using Data Envelopment Analysis. In: Abramowicz, W., Mayr, H.C. (eds.) Technologies for Business Information Systems, pp. 121–132. Springer, Dordrecht (2007)
[20] Luo, W., Strong, D.M.: A Framework for Evaluating ERP Implementation Choices. IEEE Transactions on Engineering Management 5(3), 322–333 (2004)
[21] McDonald, P., Giles, S., Strickland, D.: Extensions of Auto-Generated Code and NOSTROMO Methodologies. In: Proc. of 19th Int. Forum on COCOMO, Los Angeles, CA,
[22] Parthasarathy, S., Anbazhagan, N.: Evaluation ERP Implementation Choices Using AHP. International Journal of Enterprise Information Systems 3(3), 52–65 (2007)
[23] Plaza, M., Rohlf, K.: Learning and Performance in ERP Implementation Projects: a Learning-curve Model for Analyzing and Managing Consulting Costs. Journal of Production Economics 113, 1–14 (2008)
[24] Rettig, C.: The Trouble with Enterprise Systems. Sloan Management Review 49(1), 21–27 (Fall, 2007)
[25] Stamelos, I., Angelis, L., Morosio, M., Sakellaris, E., Bleris, G.: Estimating the Development Cost of Custom Software. Information & Management 40, 729–741 (2003)
[26] Stensrud, E.: Alternative Approaches to Effort Prediction of ERP Projects. Inf. & Soft. Techn. 43(7), 413–423 (2001)
[27] Stensrud, E., Myrtveit, I.: Identifying High Performance ERP Projects. IEEE Trans. Software Engineering 29(5), 398–416 (2003)
[28] Summer, M.: Risk Factors in Enterprise Wide Information Systems Projects. In: Special Interest Group on Computer Personnel Research Annual Conference Chicago, Illinois, pp. 180–187
[29] Taudes, A., Feurstein, M., Mild, A.: Options Analysis of Software Platform Decisions: a Case Study. MIS Quarterly 24(2), 227–243 (2000)
[30] Valerdi, R.: The Constructive Systems Engineering Cost Model (COSYSMO), Ph.D Dissertation, University of Southern California, Los Angeles, USA (May 2005)

[31] Verhoef, C.: Quantitative IT Portfolio Management. Science of Computer Programming 45(1), 1–96 (2002)
[32] Wieringa, R.: Requirements Engineering Research Methodology: Principles and Practice. In: Tutorial at the Int'l Conference on Requirements Engineering (RE 2008), Spain (2008)
[33] Wohlin, C.: Experimentation in Software Engineering. Springer (2000)
[34] Yin, R.: Case Study Research, Design and Methods, 3rd edn. Sage Publications, Newbury Park (2002)

The Influence of Culture and Leadership on Cost Estimation

Khaled Hamdan[1], Boumediene Belkhouche[2], and Peter Smith[3]

[1] UAE University, Al Ain, UAE
khamdan@uaeu.ac.ae
[2] CIT, UAE University, Al Ain, UAE
b.belkhouche@uaeu.ac.ae
[3] University of Sunderland, Sunderland, UK
peter.smith@sunderland.co.uk

Abstract. Culture and leadership factors play an important role in software development and cost estimation. We discuss the many dimensions of culture and leadership and their impact on cost estimation in software development. We conducted a survey to identify leadership and cultural factors that may influence the software development process and its associated cost. A cost estimation model incorporating these factors was developed and evaluated.

Keywords: Effort estimation, Leadership, Team culture, CBR, Ontology.

1 Introduction

Culture and leadership impact significantly the operation of an organization [1]. The quality of the software team (i.e., capabilities of the project manager, the programmers and the analysts) is a major factor in determining the cost and quality of software products [2]. The values of the leader(s) and individual awareness of the culture of the organization are determining factors in the organization productivity. Organizational culture incorporates a set of assumptions, beliefs, and values, which guide the organization members' functions. Culture is one of the most important aspects that affect peoples' lives, their behaviours and their thinking. Culture is not an easy concept to define. It "has been aptly compared to an iceberg. Just as an iceberg has a visible section above the waterline, and a larger, invisible section below the water line, so culture has some aspects that are observable and others that can only be suspected, imagined, or intu-ited. Also like an iceberg, that part of culture that is visible (observable behaviour) is only a small part of a much bigger whole" [3]. Researchers have repeatedly shown that the lack of leadership support within a project is often a cause of the project ultimate failure [4]. The leader of an organization has an essential role to play in setting the vision that the organization should embrace to move towards.

The main hypothesis of our research is that organizational culture and project leadership are significant contributing factors in the cost of software development. For our study, we selected the Arabian Gulf States [5]. In order to test

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 223–232, 2008.

this hypothesis, a survey of software development projects within government departments in the United Arab Emirates (UAE) was undertaken. The analysis of the survey highlighted several parameters affecting cost estimation in this area. Based on this analysis, new projects were monitored to ascertain the impact of organizational culture and leadership on software effort estimation. Our ultimate goal was to develop a CBR-based cost estimation model that incorporates leadership and culture (Fig. 1) [6],[7],[8].

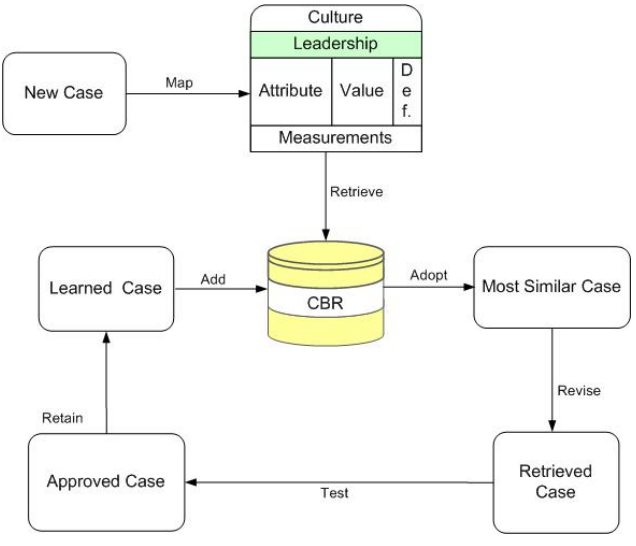


Fig. 1. The Augmented CBR

2 Leadership and Culture Parameters

Numerous models for measuring and estimating software development efforts have been proposed [1], [4]. These models have not focused on cultural issues within organizations or leadership characteristics of project managers. In an effort to derive an improved cost estimation model that is sensitive to local customs, we used a survey to identify which parameters impact the cost estimation model development. The identified parameters are categorized into seven groups (Fig. 2): Organization Line of Business, Application Type, Organization Type, Organization Culture, Project Leadership, Project Technical Environment and Year of Project Completion.

While studying these parameters we defined a Case-Based modelling process as shown in Fig. 3. This process identifies the stages to follow and the parameters that should be taken into consideration when performing software effort estimation. This process was also used in the development of the implemented system.

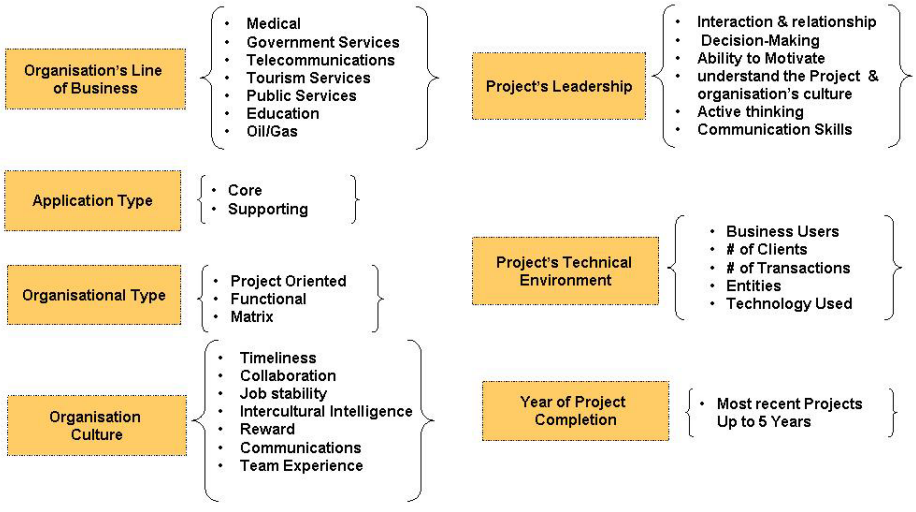


Fig. 2. Parameter Groups

Fig. 3. Proposed Software Effort Estimation Process

Various generic attributes such as personality traits, power relationship and behavior changes were observed. Figure 4 summarizes the common leadership and cultural factors we selected. Based on this figure, we elaborated an ontology for culture and leadership. The ontology establishes a common measurement protocol and provides a uniform interpretation of project parameters. Figure 5 shows the team culture ontology and Figure 6 shows the leadership ontology. The purpose of such an ontology is two-fold: (1) to guide explicitly the measurement process; and (2) to associate measurement scales with each parameter. In our survey, a 1-9 scale was used to accommodate the majority of respondents and their responses. This would also give the responses a true value of their significance. The values of the corresponding Team Culture variables are specified as: 1-3 (Low), 4-6 (Nominal), and 7-9 (High).

Six factors were used to measure leadership characteristics, whereas seven factors were used to characterize culture. For example, communication was measured based on team and leader communication skills.

"Timely" means respecting time and the individual understands the general perception of time (event or relationship). If it is 95% of the time, then the value is clearly high. Low means that the individual slacks or has frequent absences. Nominal is when the individual frequently comes late to work or to a meeting. Collaboration (impersonal relations) means the leader does not allow personal

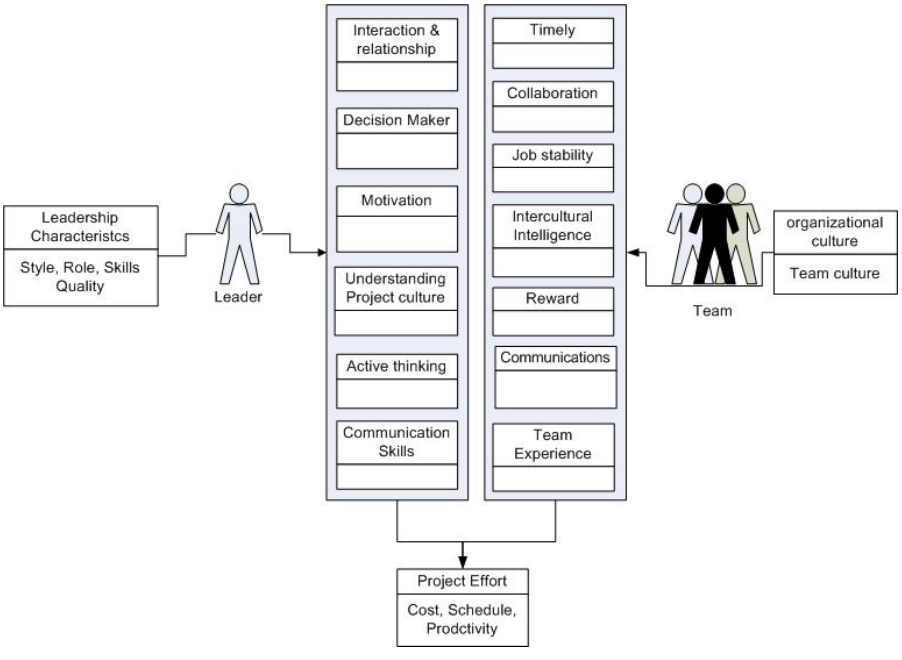


Fig. 4. Culture and Leadership Parameters

relationships to affect work. If this is 95% of time, then the value is high. Low means that the leader allows personal relationships to dominate work. Nominal is when the leader allows some personal relationships to affect work. Job stability means that the leader is a team player and holds no grudges against team members and his relationship is based on mutual trust and respect. If this is 95% of time, then the value is high. Low means that leader takes matters personally and has no trust. Nominal is when the leader holds some grudges against other members and teams. Intercultural Intelligence (impersonal relations) represents the ability to understand another culture's world view. If it is 95% of the time the leader understands others feelings, values, and goals and his ability to understand other culture world-views, and the value is high. Low means that the leader doesn't understand much about other people's feelings, values, and goals. Nominal means the leader is ignorant of other people's feelings, values, and purposes. Reward (Incentives) means the leader encourages and supports team professional development and rewards. If it is 95% of time, then the value is high. Low means the leader does not encourage or support team professional growth or rewards the team for achievements. Nominal means the leader provides some encouragement, supports the team's professional growth, and rewards some of the team's achievements.

Decision making means the leader encourages team members to communicate effectively. If it is 95% of the time, the team uses past experience to develop current or new projects and the value is high. Low means the leader doesn't

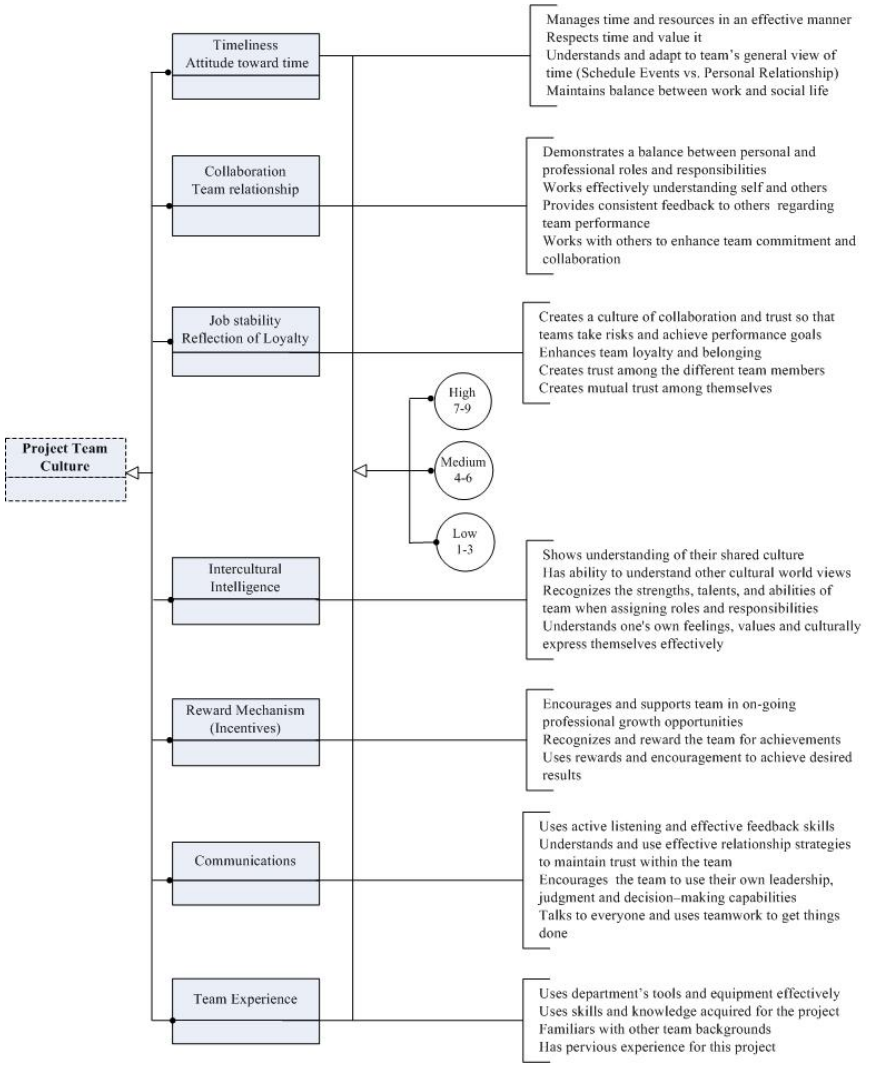


Fig. 5. Team Culture

allow any team members to use their own leadership, or decision making capabilities. Nominal means the leader allows some of team members to use their own leadership, decision making capabilities Team experience means the team works on similar projects and has the skills and knowledge. If it is 95% of time, then the value is high. Low means the team has never worked on similar projects nor skills or knowledge of the new project. Nominal means the team has worked on some similar projects and has some experiences and knowledge.

The leadership ontology categorizes several attributes (Fig. 6). Interaction (Behaviour) and relationship with team members means that the leader creates

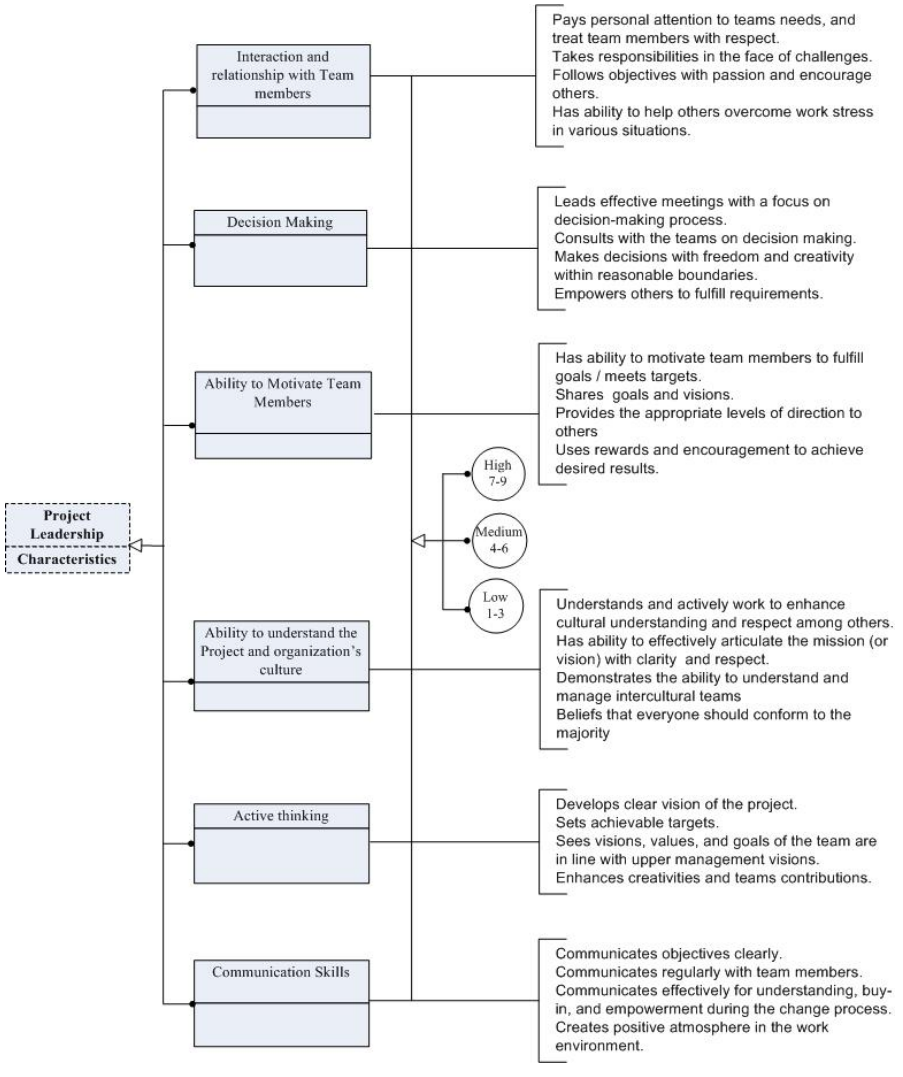


Fig. 6. Project Leadership

learning experiences, and treats team members with respect. If it is 95% of time, then the value is high. Low means the leader doesn't appreciate any of the team members' work. Nominal means the leader doesn't appreciate many of the team members' work. Decision Making of Leadership means the leader creates the right decisions and consults with teams about the organization's direction. Low means the leader doesn't make many proper decisions and doesn't react quickly to make decisions about the organization's problems. Nominal means the leader makes moderate decisions and consultations with teams and the reaction in making decisions to the organization's problems are slow or non-existent. The Ability

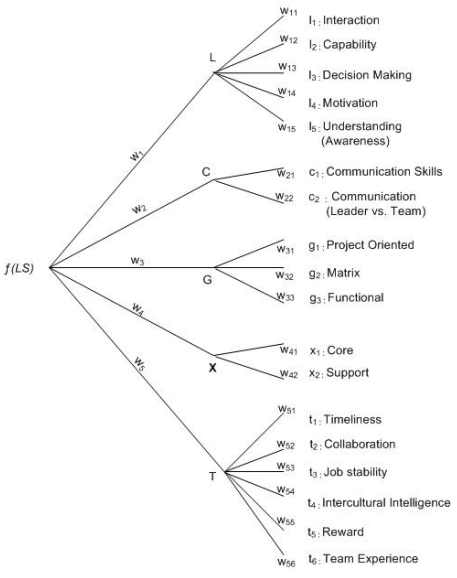


Fig. 7. Leadership factors tree

to Motivate Team members means that the leader shares goals and appropriate instructions and support. If it is 95% of time, then the value is high. Low means that the leader doesn't share many goals and appropriate levels of direction and support. Nominal means that the leader shares some of the goals with low encouragement to achieving and using some appropriate levels of direction and support are used.

The ability to understand the Project and the Organization's Culture means that the leader is able to understand and manage multicultural teams. If it is 95% of time, then the value is high. Low means that the leader doesn't have understanding of or management of intercultural teams. Nominal means that the leader has moderate understanding and managing intercultural teams. Active thinking means that the leader enhances team contributions and sets feasible target. If it is 95% of time, then the value is high. Low means that the leader doesn't enhance some of the team contributions. Nominal means that the leader has moderate enhancement, and medium team contributions feasible target. Communication Skills means that the leader uses communication among team members effectively. If Communicating regularly is 95% of time, then, the value is high. Low means communication is rare and has no effective feedback on team issues. Nominal means that the leader has some assisting and communication among team members.

Trying to combine leadership characteristics with other variables is difficult and involves quantitative measures of capability. The completeness property is important for the identification of the essential profile factors and for these to be incorporated into the profile description. One issue we are still addressing is how

to use profile theory to describe leadership. For example, using the leadership factors tree (Fig. 7), we express the leadership (LS) function as follows:

$$f(LS) = \{(\epsilon_1, L, \omega_1), (\epsilon_2, T, \omega_2, (\epsilon_3, G, \omega_3), (\epsilon_4, C, \omega_4), (\epsilon_5, X, \omega_5)\}$$

Where:

- L: Leadership characteristics, such as style, power, capability, traits, and skills
- T: Team characteristics, such as culture, knowledge, personal competencies
- G: Organizational type, such as project-oriented, functional, or matrix authority
- C: Communication skills, in both channels (leader vs. team culture)
- X: Project complexity, such as core or support systems
- ϵ_i: Factor existence, such as = 1, non existence = 0; where is ϵ_1, ϵ_2, ϵ_3, ϵ_4, and ϵ_5 are the factor existence for leader characteristics, team culture, organizational type, communication skills, and complexity, respectively.
- ω_i: Total weight of sub-factor(s) weight is divided equally in approximation ad hoc cases or based on importance or priority, where ω_1, ω_2, ω_3, ω_4, and ω_5, are the weights for leadership characteristics, team culture, organizational type, communication skills and complexity, respectively.

3 Implemented System

We implemented a software tool, called SEEOS (Software Effort Estimation Ontology System) that supports the application of an analogy based method (Fig. 1 and Fig. 8). The tool provides a flexible interface that allows users to experiment with different project characteristic options. The main functions of SEEOS are the following: defining comprehensive attributes for a project, defining attributes characteristics and measurement protocol, providing choice of options to be considered such as culture factors and leadership, determining which attributes are available to provide better accuracy, and generating most similar projects for the required estimate. SEEOS consists of three subsystems: the analogy subsystem to find the most similar projects, the online subsystem used by different organizations to input projects data, and the bootstrap subsystem to validate the project result. Figure 8 shows the user interface of SEEOS. The left side panel shows the project's entities along with their attributes, descriptions and values. The right side panel shows the selected entities by the project manager(s) to be estimated. The model was tested on a number of governmental development projects in order to determine its accuracy and appropriateness. Experiment results were then analyzed. Our estimated results showed a good fit to actual data.

As the system was designed to provide an environment for testing the feasibility and validity of the proposed model, the developed version required further improvement. One issue that we faced was clustering. To be able to search for all possible feature subsets the system needs to cluster cases by business domain and complexity. Because the particular projects under investigation were from various domains and shared few projects, clustering was somehow difficult. For

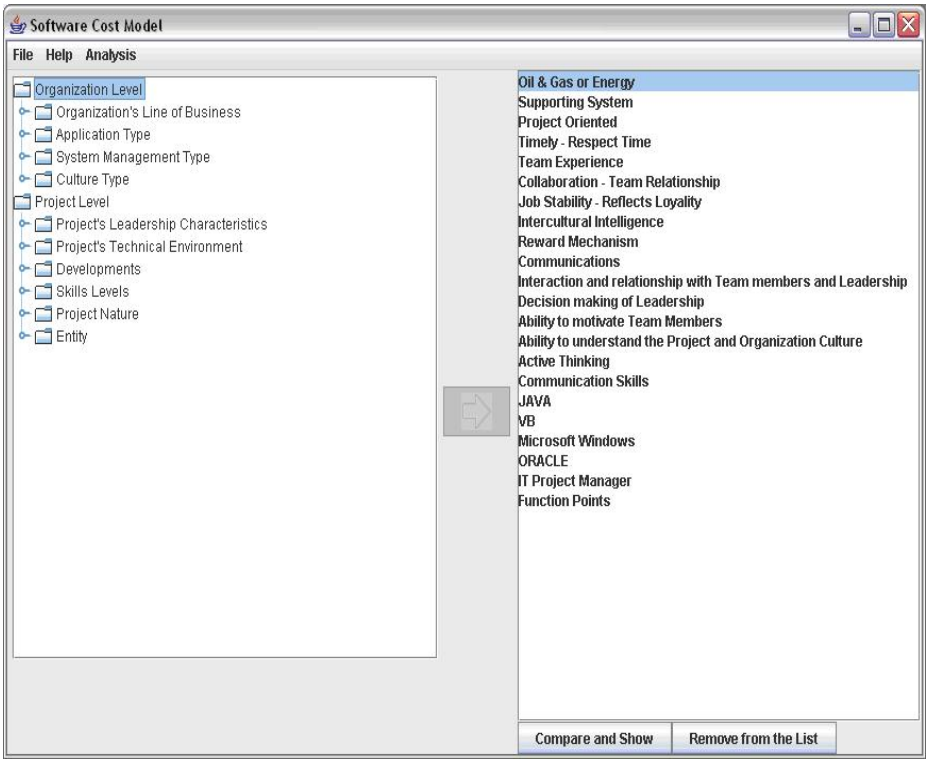


Fig. 8. SEEOS User Interface

example, it was difficult to measure the similarity of core systems in a different domain. A possible strategy would be to refine this notion of core systems by categorizing them according to application domains.

4 Conclusion

In the Gulf States, culture and leadership play a bigger role in affecting work performance. From the questionnaire and interviews, we concluded that software cost estimation in the Gulf often does not use accepted cost models. Due to the lack of a uniform protocol, variations in measurement cost among projects were noticed. To this effect, we categorized various factors affecting cost estimation and we developed an ontology to support an unambiguous interpretation of these factors. We also integrated culture and leadership in the CBR model. The inclusion of leadership and culture in the cost estimation model constitutes an enhancement and refinement. It also offers a possible enhancement to current models which do not take leadership and cultural backgrounds into account. In our research, two models, one with culture and one without culture have been used. Experiments to compare the effectiveness of these two models are still being carried out.

References

1. Kitchenham, B., Pfleeger, S.L., Fenton, N.: Towards a Framework for Software Validation. IEEE Trans. on Software Engineering 21(12), 929–944 (1995)
2. Boehm, B.W.: Software Engineering Economics. Prentice-Hall, New Jersey (1981)
3. The Peace Corps Cross-Cultural Workbook, p. 10 (2008)
4. Futrell, R., Shafer, D., Shafer, L.: Quality Software Project Management. Software Quality Institute (2002)
5. Hamdan, K., Abu Sitta, F., Moses, J., Smith, P.: An Investigation into the Gulf States Government Approaches to Software Development and Effort Estimation. In: BCS 10th International Software Quality Conference, pp. 111–126. BCS Press, London (2005)
6. Shepperd, M., Jorgensen, M.: A Systematic Review of Software Development Cost Estimation Studies. IEEE 33(1), 33–53 (2007)
7. Shepperd, M., Schofield, C.: Estimating Software Project Effort Using Analogies. IEEE 23(12), 736–743 (1997)
8. Angelis, S.: A Simulation Tool for Efficient Analogy Based Cost Estimation. Empirical Software Engineering 5, 35–68 (2000)

Portfolio Control – When the Numbers Really Count

Frank W. Vogelezang

Sogeti Nederland B.V., Lange Dreef 17, 4131 NJ Vianen, The Netherlands
frank.vogelezang@sogeti.nl

Abstract. Most IT-metrics and metrics related research focus on single applications or single projects. From a research point of view this is understandable. For most IT consuming organisations the results of single applications or projects is less relevant. To those kind of organisations the performance result of the whole application- or project portfolio is a more relevant focus because the governance is positioned on the portfolio level rather than the project level. In this paper we present some of our experiences in using IT-metrics for portfolio control.

Keywords: Portfolio Control, Portfolio Management, Scope Management, Project Management, Application Management, Functional Size Measurement, Governance.

1 The Difference between Projects and Portfolios

There is a difference between the metrics used to control single applications or projects and metrics used to control portfolios. Portfolio control deals with the value – usually expressed in financial terms – of the portfolio. Project or application control focusses on more operational entities like effort, duration and staffing levels.

Portfolio control is more than an extension of project and application control to deal with multiple occurrences. To control a portfolio one cannot simply use the same metrics as for a single application or project and extrapolate them to the whole portfolio. Portfolio control addresses different needs of information on the portfolio level. It relies on information that is gathered and used for operational application and project control purposes.

In application and project control the focus usually is on development of the IT assets, where portfolio control covers a much wider area of attention including maintenance efficiency, strategic IT choices, cost reduction and the valuation of IT assets.

2 Portfolio Control

Portfolio Control is not a commonly accepted knowledge area within the IT domain. The portfolio realm of thought has its roots in the finance domain [1]. For this paper the following definition of Portfolio Control is used:

> Portfolio Control is the use of metrics of relevant portfolio aspects to support and justify management decisions about the portfolio.

Portfolio Control is an important toolset for IT Portfolio Management, or in a wider context, Information Management. Within the IT domain two categories of Portfolio Control can be distinguished.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 233–244, 2008.

The first is Application Portfolio Control, to support and justify active management of the coherent set of applications which support the realisation of the business processes [2], usually referred to as Application Portfolio Management. Although the larger part of the IT budget is spent on support and maintenance of the current applications – with MOOSE[1] levels of 60% and higher – controlling it is seen as an IT task in most companies [3].

The other is Project Portfolio Control, to support management decisions about the changes to be made to the current portfolio. This part of the IT budget is in most cases controlled by the financial management.

Quite often project portfolio management is embedded in the strategic decision chain, while application portfolio management is seen as an operational issue with less strategical impact [4]. Both categories will be dealt with in this paper, although the emphasis will be on Application Portfolio Control.

3 Application Portfolio Control

Application Portfolio Control does not offer a single metric to support the management of the application portfolio. Application Portfolio Management has too many aspects to be dealt with to fit into a single meaningful metric. This is similar to describing the functions of a city in square centimeters on a map. The fact that Application Portfolio Management requires understanding of various, mostly technical, aspects is probably the reason that this is usually positioned in the IT domain.

To compare different elements of a portfolio all elements should have one metric in common that can be used to normalise all value aspects of a portfolio. For IT portfolios a functional size measurement like function points can be used.

3.1 The Normalising Metric

As mentioned, the area on a map is not a good metric to describe the functions of a city. It is very useful though to relate aspects of the city in different areas. In application portfolios the functional size serves a similar purpose. A big hurdle in applying functional size measurement as normalising metric is the fact that is a labour intensive activity. A detailed function point count of an application portfolio can easily cost hundreds of person hours of skilled function point specialists if all the documentation is available and much more if the portfolio is poorly documented.

To support management decisions it is often not necessary to have the precision of a detailed function point count. For larger portfolios approximation methods are available to balance the required counting speed with the necessary precision [5].

In Service Oriented Architectures applying function points can be difficult. To be able to measure the size of the components the COSMIC method can be used [6]. If desired, the acquired size in COSMIC Function Points can be converted to a size in traditional function points [7]. This should be done with care, since this conversion has a limited validity.

With the normalising metric in place all relevant aspects to support and justify active management of the application portfolio can be put into the perspective of the amount of functionality an application offers to support the realisation of business

[1] MOOSE continued spending on **m**aintenance and **o**ngoing **o**perations, **s**ystems and **e**quipment.

processes. Currently the function point metric is the most common and most useful metric for comparative purposes [8].

3.2 Service Level

Different service levels can have an enormous impact on the (normalised) distribution of available maintenance capacity over the different elements of an application portfolio. This is one of the things that is often forgotten in comparing the amount of staff needed to support and maintain different parts of the application portfolio. This is best illustrated by an example of two different applications:

- *Electronic Banking Service*
  Functional size: 1,000 function points
  Needs a 24/7 service level for an availability of 99.95%
  This SLA needs 168 man-hours per week at a cost level of € 125,= on average
  Weekly cost level: (168 x 125) / 1,000 = 21 €/FP
- *Management Reporting Application*
  Functional size: 9,000 function points
  Needs a next day service level for an availability of 50% during office hours
  This SLA needs 20 man-hours per week at a cost level of € 95,=
  Weekly cost level: (20 x 95) / 9,000 = 0.21 €/FP

So the difference in the cost of a 24/7 service level and a next day service level is in the order of magnitude of 100. If the average cost level over the whole portfolio is about 2 €/FP then the first application will be underfunded about ten times and the second will be overfunded about ten times if the service level is not taken into account.

Differentiating the cost of different service levels can be of great help to the business to decide what type of service level is needed for the applications within the portfolio. A lower service level has a high potential of reducing the cost of maintenance and support for an application portfolio. This cost aspect has to be balanced against the cost aspects of the application not being available. Mature organisations determine the service level in a business case as an insurance premium for consequential loss resulting from inavailability of the application.

3.3 Application Complexity

Application complexity is a serious factor to take into account. The application complexity is often expressed in terms of cyclomatic complexity of the source code. Cyclomatic complexity measures the amount of branches in the control flow of an application. Perfectly structured source code, with no branches of any kind, has a cyclomatic complexity of 1. In practice, levels of less than 10 are considered to have a low complexity. Levels of greater than 20 are considered to have a high complexity since the source code is poorly structured. Levels between 10 and 20 are considered to have an average complexity.

The software maintenance productivity doubles between each complexity category. So if the complexity is reduced from high to low the software maintenance productivity can improve about four times [8]. This experience is usually a good argument for a business case to restructure applications without adding new functionality.

3.4 Application Age

The age of an application has an impact on the normalised cost and effort to support and maintain this application. The cost of support and maintenance together over time tends to a pattern that resembles a classical bathtub [9].

The cost is relatively high at the beginning because of set-up problems and teething troubles. When these issues are solved the support and maintenance cost stabilise. Over the years the combined effects of accumulating maintenance and the heritage of retraced design decisions will show in the form of increasing cost levels. At first the levels will increase gradually, but eventually the cost levels will rise increasingly faster until the cost levels go sky-high and the application becomes expensive legacy.

Based on this relation it can be predicted when an application should be replaced in financial terms. This point in time can be calculated from the exponential trend lines of the cost curve. In the example on the next page – showing data from the Dutch pension firm Cordares [9] – the exponential trend line is still decreasing in the first six years of operation. From year 8 on the exponential trend line starts to increase rapidly.

The observed cost can be approximated to a cost function. The ideal replacement period can be calculated by finding the minimum of the cost function divided by the number of years in operation. For Cordares systems, which have a relatively low cost for maintenance and support, the ideal replacement period is 11 years.

To be able to use this metric properly requires a good discipline in the registration of cost and hours spent, so it is not advised for relatively immature organisations.

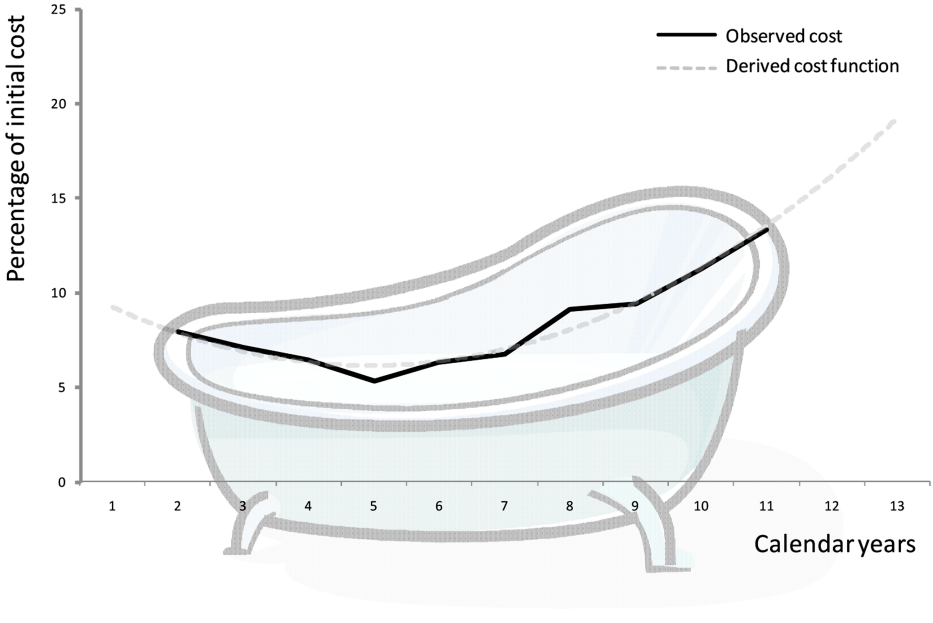


Fig. 1. Cost for maintenance and support over time

3.5 Application Size

The size of an application is also a factor to take into account. Results from the SPR database suggests that the software maintenance productivity improves about 10% if the size is about 10 times smaller [8].

Especially in portfolio's that are very heterogeneous in size this might be worth to take into account. The typical portfolio consists of a relatively large number of small to medium size applications (all less than 4,000 FP) and a few core applications that exceed 10,000 FP in size. For such a portfolio it is usually sufficient to correct only the core application(s) for this effect.

In portfolios with packaged applications attention is necessary what size will be used to normalise the value aspects. For different types of packaged applications different approaches should be used to determine the right size of a packaged application.

- For packaged applications in which the functionality is implemented by parameterisation or configuration the size of the offered functionality needs to be determined. This size is usually much smaller than the total size of the packaged application. Examples of these kind of packaged applications are SAP, Oracle E-Business Suite and Microsoft Dynamics. To determine the size of this kind of applications we have developed an estimation method partly based on rules of thumb [5].
- For packaged applications in which the functionality that is offered to the end-user can be influenced by turning parts of the application on or off also the size of the offered functionality needs to be determined. Determining this size is usually more straightforward than with the previous category. Examples of these kind of packaged applications are tools that support ITIL processes, like Applix or Peregrine, but can also be configured to support only selected processes.
- For packaged applications that are implemented as-is and only have a limited amount of configuration options, like office automation, the functional size is not a meaningful metric. For these kind of applications there is no relation between the offered functionality and the relevant portfolio characteristics. To be able to use the functional size as a normalising metric one can calculate a fictitious size by backtracking from the budget [5]. In several larger application portfolios we calculated values of around 200 function points for this type of applications. This is only relevant for metrics that relate to staffing levels. For asset-type metrics the functional size should not be used.

3.6 Number of Applications

Application portfolios have a natural tendency to grow, both in size and in number. The size growth can be managed by a good replacement policy – or less politely: kill management – based on the principle discussed in section 3.4. When an application is replaced care should be taken in only replacing still required functionality, or else the natural growth rate will continue.

When the application portfolio is not managed effectively duplicate functionality enters the portfolio. When more than one application is used for the same purpose, this is usually not cost effective. Very often duplicate functionality is the result of a lack of information about available functionality within the current portfolio. By creating and sustaining a good inventory of available functionality the growth rate of the application portfolio can be slowed down.

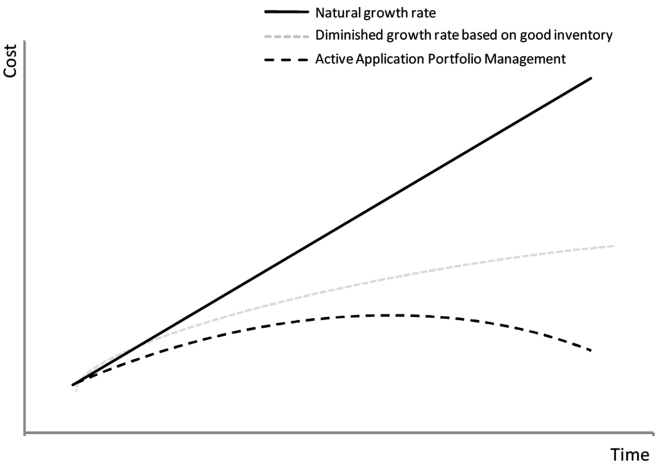


Fig. 2. Application Portfolio cost over time in different scenarios

By using active application portfolio management the growth can not only be slowed down by preventing duplicate functionality to enter the portfolio, but even be reversed by actively removing unwanted duplicate functionality. This kind of program is usually a long-winded road, but it can lead to a substantial cost reduction [2]. Sogeti has developed a five-step cyclic approach for this type of application portfolio management programs [10]:

- *Scope & Approach*
  In most organisations an evolutionary scope and approach is to be preferred over a revolutionary large-scale implementation with a full-blown tool implementation. Especially the implementation of tooling for this type of program is very much resistance prone.
- *Portfolio Analysis*
  The first part of this step is to make or complete the inventory, not only of the applications in the portfolio, but also of the information management needs which the applications are meant to support.
  The second part of this step is the real analysis, to determine the added value for the business and the overlap in functionality.
- *Lifecycle Management*
  When the portfolio has been analysed, all applications must be positioned in their respective life cycles, and should be assessed in view of architecture and sourcing principles. This results in a maintenance strategy for each application.
- *Implementation*
  Quick wins in the maintenance strategies must be taken and the results of the program must be made visible to the whole organisation.
- *Consolidation & Evaluation*
  The Application Portfolio Management program must be consolidated within the service and architecture processes of the organisation to make sure the portfolio growth rate is under permanent control.

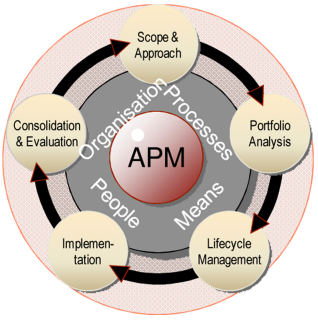


Fig. 3. Sogeti Application Portfolio Management Approach

This instrument for portfolio control is very useful for organisations that have a decentralised (financial) structure or have inherited a fragmented application portfolio from mergers and acquisitions.

3.7 Architecture Fit

The amount of support effort needed reflects whether functionality, application and infrastructure comply with the architectural guidelines. Non-compliance introduces the need for extra non-standard knowledge and capabilities and thus is not cost effective.

Architecture cannot be measured in terms of a ratio or absolute scale and the effect of the architecture fit cannot be used in calculations. The architecture fit can be expressed in terms of an ordinal scale to be able to rank applications on their architecture fit. Usually a simple scoring table is sufficient to review applications on architectural guideline compliance.

This metric is very useful in combination with an Application Portfolio Management program to support decisions about maintenance strategies.

4 Project Portfolio Control

Just like Application Portfolio Control does Project Portfolio Control not offer a single metric to support the management of the project portfolio. Since Project Portfolio Control is usually positioned in the Finance domain, technical metrics should and can be avoided. To compare different elements of a portfolio all elements are expressed in terms of cost and time.

4.1 The Cost of Time to Market

The cost of an IT project is bound by at least two laws: the hydraulic software law for the time-effort trade-off and the cost of development function [11].

Using these laws on a portfolio level can show the effect of managing the complete project portfolio on time to market [12].

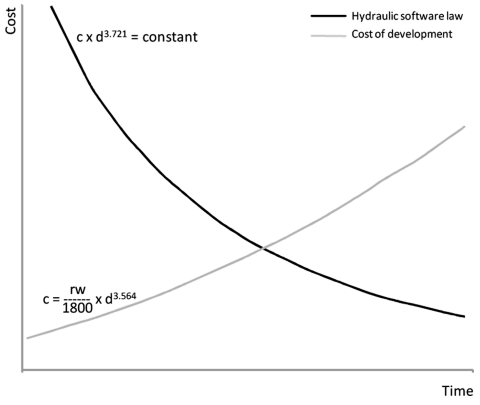


Fig. 4. Laws governing the cost of an IT project

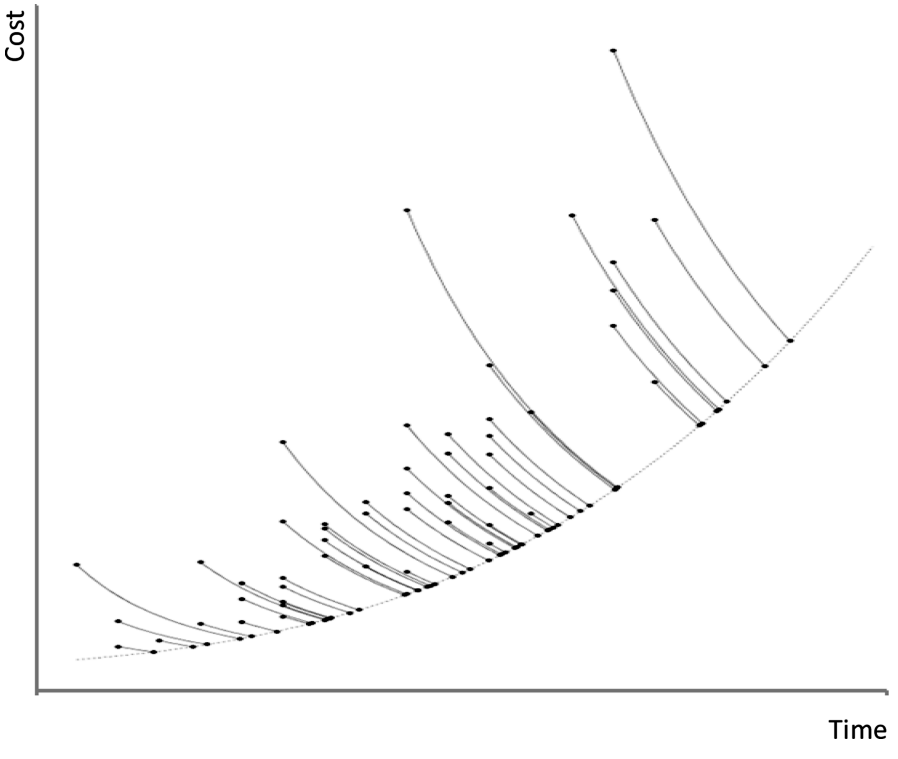


Fig. 5. The cost effect of managing a project portfolio on time to market

If all projects could be managed according to the cost of development function a reduction of up to 36% can be achieved on a portfolio level. This cost reduction is not achievable in practice. There will always be projects that need to be managed on time to market, either to gain competitive advantage or to comply with legislative requirements.

Using this principle can still save a large amounts of cost and resources that can be spent on projects where the cost of time to market is spent best.

4.2 Benchmarking Project Proposals as Risk Assessment Filter

The same laws can be used to filter out project proposals that have a high degree of risk of being poorly estimated. The figure on the next page represents about 200 projects, at a total cost of a little under a billion US dollar. These are completed projects with actual cost and duration. The results of this portfolio, which is assembled from a number of different financial institutions, can be used as a benchmark for project proposals.

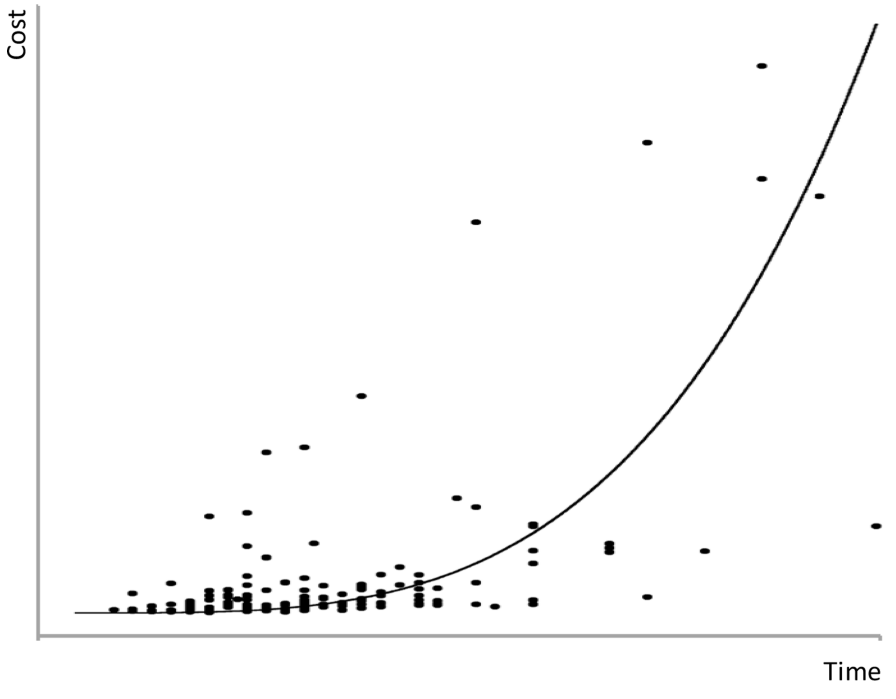


Fig. 6. The cost and time relation of a project portfolio

When project proposals deviate too much from the benchmark formula, this is a serious indication that there might be something wrong with the project's estimate. If a proposal is near the benchmark formula then there is a low, acceptable risk that this project has been wrongly estimated [13].

This benchmark formula can be used as a quick first assessment of the quality of a large number of project proposals, so that enough attention can be devoted to the project proposals that either need management attention or should not be approved.

4.3 Project Value Versus Project Risk

An important dimension of portfolio management is the value of a project as part of the portfolio. Value alone is not a good enough decision parameter, it should always be combined with risk. Both value and risk can be expressed in a number of different

ways. One of the more objective techniques is to use the Net Present Value (NPV) of a project for a given timeframe, usually 5 years. From the Finance perspective this technique is more reliable than, for instance, ROI or break-even calculations.

For different scenario's the NPV is calculated and from this the mean NPV is derived. The fraction of the mean NPV over the Capital Expenditure (Profitability Index), can be used as a measure for the value of a project.

Risk can be expressed as the fraction of difference between the minimum and maximum NPV over the mean NPV. This metric expresses the tendency of the project to deviate from the expected value. Combining both the value and the risk metric gives a decision matrix as shown below.

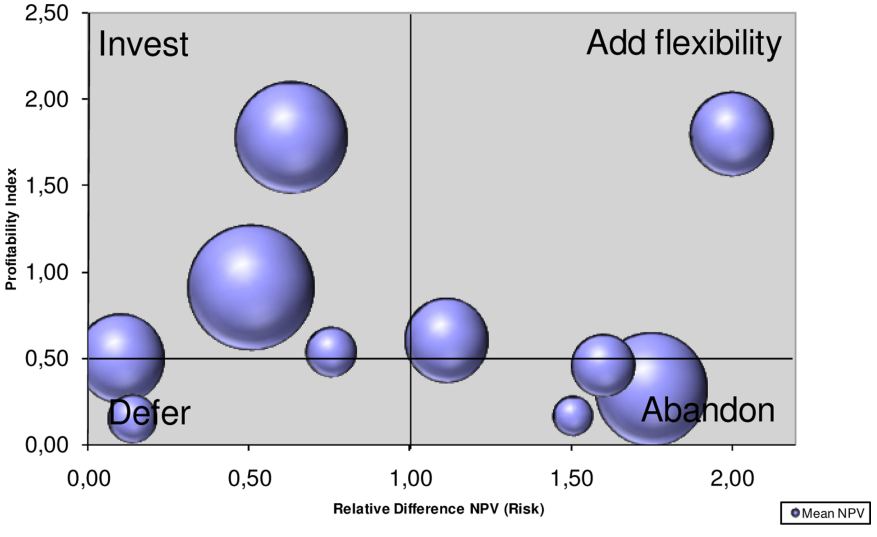


Fig. 7. Financial metrics for portfolio selection

A treshold level can be used as a selection criterium for the value based selection. Projects below that treshold should be either deferred or abandoned. This treshold level can either be fixed, or based on the available budget for capital expenditure. If the available budget is limited, the treshold level for the Profitability Index will rise.

The same goes for the risk level. Depending on the environment in which an organisation operates and which stakeholders it needs to satisfy, the organisation can choose a more defensive, or a more aggressive, risk treshold.

5 Tools to Support Portfolio Control

For some reason, tooling is one of the first aspects that is brought up in relation to portfolio control. Probably because of a perception that the amount of data needed for portfolio control is huge and needs automated support. This is one of the common pitfalls for introducing portfolio management and portfolio control in an organisation.

The first things that should be considered is which elements of the portfolio are important for the portfolio governance.

Portfolio control is more than an extension of project and application management to deal with multiple occurences. It addresses different needs on the portfolio level, but it relies on information from the project and application level. Ideally, tooling for portfolio control should integrate with project management, financial and support tooling. But to get portfolio control started it is more important to focus on getting the right data available, than on a good tool to facilitate this [14].

Portfolio control enables the management of the project portfolio to maximize the contribution of projects and applications to the overall success of the portfolio, meaning [15].

- aligning portfolio elements to the corporate strategy and goals
- creating and maintaining an overall positive cash flow for the portfolio
- aligning the effective use of resources – both human and material resources.
- balancing short-term results with long-term cost effectiveness
- maintaining the pipeline: continuing, delaying, or terminating portfolio elements

Most project management, financial and support tools were not designed to hold the ranking data or to display it in ways that facilitate portfolio decisions by the governance. For this, a specifically designed portfolio management tool is needed. In addition, some recognized decision support tools have been optimized for supporting portfolio management. The portfolio management process is not unlike that used in selecting items for an investment portfolio. In fact, the IT portfolio is an investment portfolio: you are investing in IT assets with the objective of maximizing the return for your organisation [14].

6 Conclusion

Enough techniques from different competences are available to achieve the desired amount of control over an IT portfolio. With the use of these portfolio control metrics gut feeling can be exchanged for serious quantitative portfolio management.

References

1. McFarlan, F.W.: Portfolio Approach to Information Systems, Harvard Business Review (September-October 1981), http://harvardbusinessonline.hbsp.harvard.edu
2. van der Kleij, P., Hakvoort, K.: APM – More or less the same. In: 13th Dutch ITSMF congress, Nijkerk, The Netherlands, October 27-28 (2007), http://www.itsmf.nl
3. Bartels, A.: Defining the MOOSE in the IT Room – Measuring your IT Spending like others do, Forrester Best Practices, October 18 (2005), http://www.forrester.com
4. Symons, C., Orlov, L.M., Bright, S., Brown, K.: Optimizing The IT Portfolio For Maximum Business Value, Forrester, September 30 (2005)
5. Rispens, M.A., Vogelezang, F.W.: Application Portfolio Management, the basics – How much software do I have. In: Proceedings of the 4th Software Measurement European Forum (SMEF 2007), Roma, Italy, May 9-11 (2007), http://www.iir-italy.it/smef2007

6. Santillo, L.: Seizing and Sizing SOA Applications with COSMIC Function Points. In: Proceedings of the 4th Software Measurement European Forum (SMEF 2007), Roma, May 9-11 (2007), http://www.agilemetrics.it/risorse.html
7. van Heeringen, H.: Conversion of Functional Size – FPA ↔ COSMIC, VIII Congreso Anual de la Asociación Española de Métricas de Sistemas Informáticos, Madrid, October 1 (2007), http://www.aemes.org
8. Jones, C.: Applied Software Measurement – Global Analysis of Productivity and Quality, 3rd edn. McGraw-Hill, New York (2008), http://www.spr.com/catalog
9. Labrujere, G., de Weme, H., van der Wurff, A.: Life Expectancy – Managing the IT Portfolio of a Pension Firm. In: Proceedings of the 1st IEEE Computer Society Conference on Exploring Quantifiable Information Technology Yield, Amsterdam, March 19-21 (2007), http://www.cs.vu.nl/equity2007
10. van der Kleij, P.: De weg naar een gezond applicatielandschap (Dutch-only), APM White Paper, Sogeti (September 2007), http://go.sogeti.nl
11. Putnam, L.H., Myers, W.: Measures for Excellence. Yourdon Press Computing Series (1992), http://www.qsm.com
12. Verhoef, C.: Quantifying the effects of IT-governance rules. Science of Computer Programming 67, 247–277 (2007), http://www.elsevier.com/locate/scico
13. Verhoef, C.: Quantitative IT Portfolio Management. Science of Computer Programming 45, 1–96 (2002), http://www.elsevier.com/locate/scico
14. Levine, H.A.: Project Portfolio Management – A practical guide to selecting projects, managing portfolios and maximizing benefits, Jossey Bass (2005), http://www.josseybass.com
15. Maizlish, B., Handler, R.: IT Portfolio Management Step-By-Step – Unlocking the Business Value of Technology. John Wiley & Sons, Chichester (2005), http://eu.wiley.com

Defining Suitable Criteria for Quality Gates

Thomas Flohr

FG Software Engineering, Leibniz Universität Hannover,
Welfengarten 1, 30167 Hannover, Germany
thomas.flohr@inf.uni-hannover.de

Abstract. A considerable number of software projects still exceed time and budget or completely fail, because the qualitative situations of these projects are not visible to the management. The problem can be resolved by monitoring the quality of project results and by steering a project at certain major points (so-called Quality Gates). At each Quality Gate the project results are checked against predefined criteria being derived from carefully chosen metrics. Many software companies use Quality Gates but unfortunately a theoretical reflection on the definition of criteria for Quality Gates is missing. This paper shows, when and how these criteria can be identified and improved over time. Our results obtained from students' software projects show, that the application of a systematic top-down approach (such as GQM) delivers better criteria and that roughly a considerable number of the criteria could be improved after experiences have been captured and reused systematically.

Keywords: Practical measurement application, Measurement acceptance, Quality Gates.

1 Introduction

Quality Gates are significant milestones and decision points within a project [1, 2]. At each Quality Gate criteria are evaluated against predefined and quality focused criteria. Based on the fulfillment of these criteria gatekeepers (which are usually part of the quality management) make a decision whether a project may proceed or not. Consequently, the quality situation of a project can be uncovered to the management and actions can be made in time.

A software company can use Quality Gates in two ways (we will refer to them as strategies):

- **Quality Gates as a quality guideline:** The same set of Quality Gates and criteria is applied to all relevant projects resulting in a comparable and at least an equal minimum quality level in all those projects.
- **Quality Gates as a flexible quality strategy:** A suitable set of Quality Gates is applied in each project to exactly meet a projects needs.

Either way, criteria have to be chosen carefully because of two reasons:

- projects delivering poor results might be proceeded
- or promising projects might be canceled.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 245–256, 2008.

Criteria are derived from metrics. Each criterion encapsulates a metric, a desired value, a predicate and the object which should to be measured. The predicate allows comparing the desired value with a measured value to determine a criterion's degree of fulfillment. In our case the predicate always delivers *true* or *false*. Nonetheless, a predicate might also deliver fuzzy values ranging from 0 to 1. We can formalize a criterion as follows:

Definition: A criterion is a tuple consisting of four parts $(m, \bar{o}, \bar{s}, p_m)$.

$m: O \to S$ is a metric, $\bar{o}$ is the actual object of measurement, $\bar{s} \in 2^S \; (|\bar{s}| \geq 1)$ is a set of desired values and p_m is a predicate. p_m is defined as follows

$$p_m : O \times P(S) \to \{true, false\}$$

$$p_m : (o, s) \mapsto p_m(o, s)$$

with the following property

$$\text{Criterion is fulfilled} \Leftrightarrow p_m(\bar{o}, \bar{s}) = true.$$

$p_m(\bar{o}, \bar{s})$ is denoted as the rating of an criterion indicating its degree fulfillment.

The following example shows a criterion:

Example:
The criterion

„All methods of the program *MyWebSchedule* must not exceed 20 lines of code (omitting comments and empty lines)."

contains the following parts

- m is a metric, mapping from the set programs to N^0 and calculating the maximum number of lines of code (omitting comments and empty lines).
- $\bar{o}$ is the program *MyWebSchedule.*
- The desired value $\bar{s}$ is 20 lines of code.
- The predicate is defined is the following way:

$$p_m : (o, s) \mapsto \begin{cases} true, & \text{falls } m(o) \leq s \\ false, & \text{falls } m(o) > s. \end{cases}$$

1.1 Outline

This paper is structured as follows: chapter 2 contains an overview of the dimensions of criteria definition as well as empirical results on the methods of definition. In chapter 3 we analyze the experience-based improvement of criteria supported by our workflow-tool NetQGate. Furthermore, we also show empirical results regarding the improvement. Chapter 4 deals with strategies for criteria improvement in order to overcome certain problems of improvement. Finally, chapter 5 contains a conclusion and an outlook.

2 Dimensions of Criteria Definition

The definition of criteria can be classified in three dimensions: time of definition, individuality of definition and the method of definition. These dimensions are explained in more detail in the following three subsections.

2.1 Individuality of Definition

A software company which applies the strategy *Quality Gates as a quality guideline* must hold their criteria in a catalogue. These criteria can only be individualized by interpreting them in the context of a project. To some degree it is also possible to ignore criteria or to add additional criteria which allows tailoring criteria in order to better meet a project's situation.

The strategy *Quality Gates as a flexible quality strategy* requires defining criteria for each project individually. Depending on the individuality of definition the following advantages and disadvantages exist:

Table 1. Advantages and disadvantages of different levels of individuality

	Fixed criteria catalogue	No criteria catalogue
Advantages	- Qualitative policy for all projects. Allows comparing the projects among each other. - Effort for criteria definition only once. - Tailoring is possible to some degree.	- Criteria perfectly match a project's situation.
Disadvantages	- Criteria are more difficult to define, because they have to be applied to all projects. - Acceptance may be lower. - Criteria must be interpreted requiring some additional resources.	- Overall, definition of criteria requires more resources, because they must be defined for each project individually. - Individual definition makes it hard to compare projects among each other.

2.2 Time of Definition

Criteria can be defined in different phases of a project: project start, planning phase and conduction phase. A software company applying the strategy *Quality Gates as quality guideline* has to define the criteria and set them as default for all projects. Since these criteria are too abstract they must be interpreted in the context of each project. The interpretation is part of the definition and also can take place in the mentioned three phases of a project.

Depending on the time of definition different advantages and disadvantages exist:

Table 2. Advantages and disadvantages of early and late time of definition

	Early definition	Late definition
Advantages	- Management has a stronger influence on the criteria. - Criteria have a stronger impact on the project, because they are defined by the management. - Criteria are longer visible to the developers.	- Criteria better meet the needs of a project, because more information is available on the project.
Disadvantages	- Definition of criteria must be conducted by the management, which often is overloaded with other tasks. - Criteria might be not suitable for a project, because not all necessary information is known that early in a project.	- It is more likely, that criteria are not taken seriously, because they are not defined by the management. - Visibility of criteria might be too short. Therefore it is more likely that criteria cannot be hold.

2.3 Method of Definition

Criteria are at best defined in a systematic way which requires deriving criteria from abstract (business) goals. Criteria then can be derived by applying the Goal-Question-Metrics (GQM) method [3] or quality models [4]. Since this is a very laborious way companies usually will avoid such methods. Criteria then will be defined by using the experiences of senior developers only. However, the experience-based approach does not guarantee a full coverage of the goals nor does it always deliver good criteria.

Criteria for Quality Gates are result-oriented and are typically located between a project's phases or iterations/increments. Depending on a certain project's phase, different metrics (and thus criteria) are typical in the corresponding Quality Gate (Figure 1).

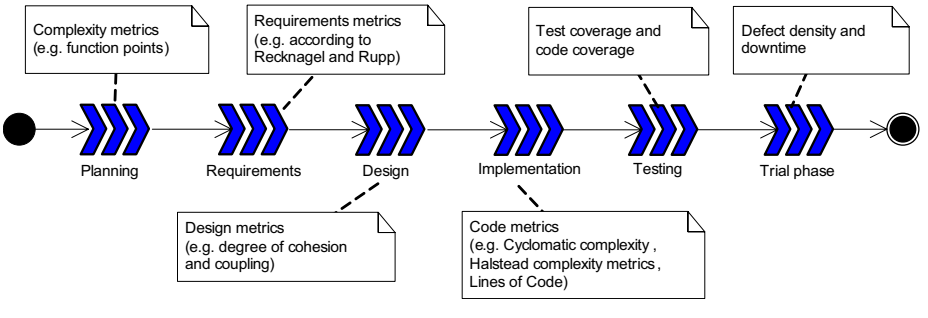


Fig. 1. Typical metrics with regard to a project's phases

We obtained data on the quality of coverage through a student project conducted at FG Software Engineering at Leibniz University of Hannover. The project was conducted in 2007/2008. Overall 16 master students (divided in two groups) participated in the project. The project aimed to construct two software modules:

A An IDE for use cases [5] realized as a service-oriented architecture.

B A change management system for use cases.

The modules were constructed in an iterative and incremental process. In last the iteration of the project, both software module were brought together to form a whole

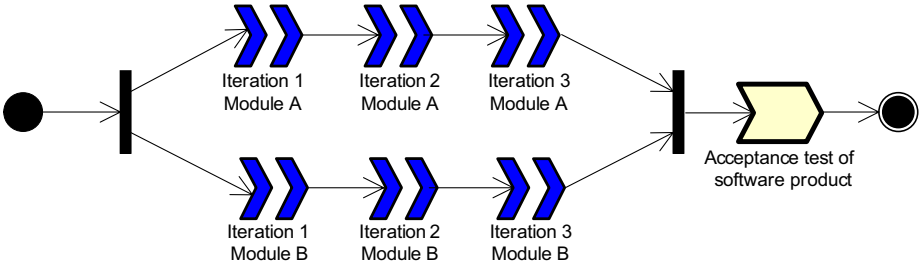


Fig. 2. Process followed in the students' project

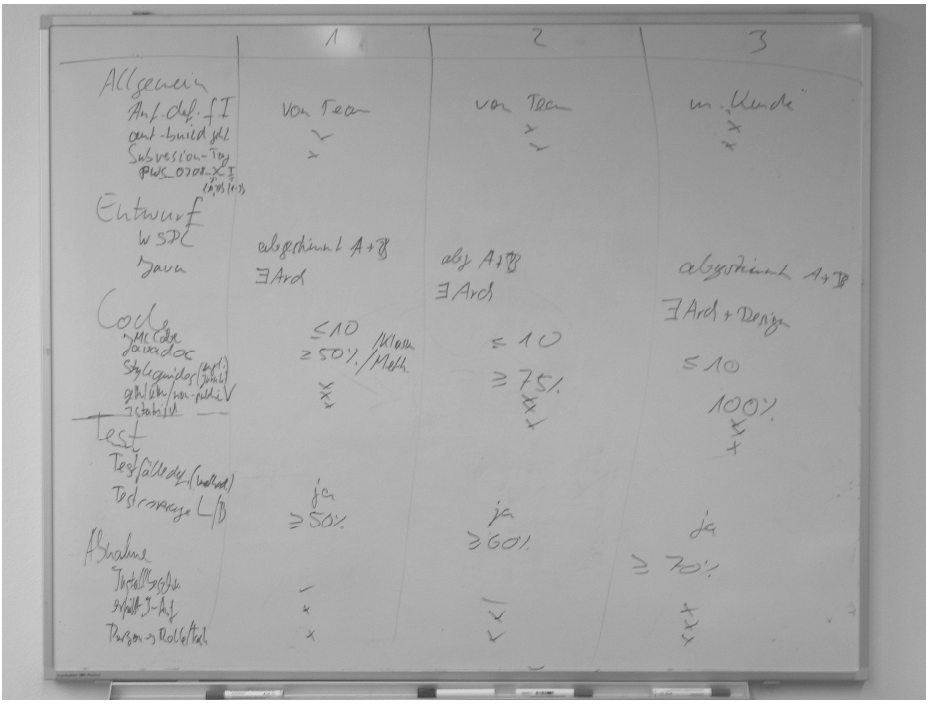


Fig. 3. Results of a experienced-based brainstorming of criteria

software product. Overall three iterations were conducted to construct each module. Each of the iterations lasted four weeks. Figure 2 depicts the process being applied in the project.

Since the project's risk was high and the quality outcome was quite uncertain, there was a strong need to monitor and steer the project often. Thus, we decided to schedule a Quality Gate between each iteration (individually for each module development track) and shortly before the acceptance test. Consequently, we had a total number of five Quality Gates. The constructed Quality Gate process followed the strategy *Quality Gates as a flexible quality strategy*, because the project situation was unique.

Firstly, criteria were defined through an experience-based approach, because some quality problems were known from a similar project. The criteria were defined by research assistants having an experience level comparable to a senior software engineer.

Criteria were collected and sorted through a brainstorming method on a whiteboard. Figure 3 shows the result of this criteria definition. Each row indicates a metric and the actual object of measurement. Each column (marked with numbers 1 to 3) shows the desired values and predicates of the according metric for one type of Quality Gate. In this way e. g. column 1 shows the criteria for the Quality Gates being conducted after the first iteration in each development track.

Since goal coverage seemed to be low, we decided to apply a goal-oriented approach (namely the GQM method) in a second step. Quality goals concerned the correctness, testability, maintainability, understandability, structuredness and the accessibility.

Figure 4 shows the maintainability branch of the GQM tree. The whole GQM tree is approximately of double size and contains a second branch covering the utility

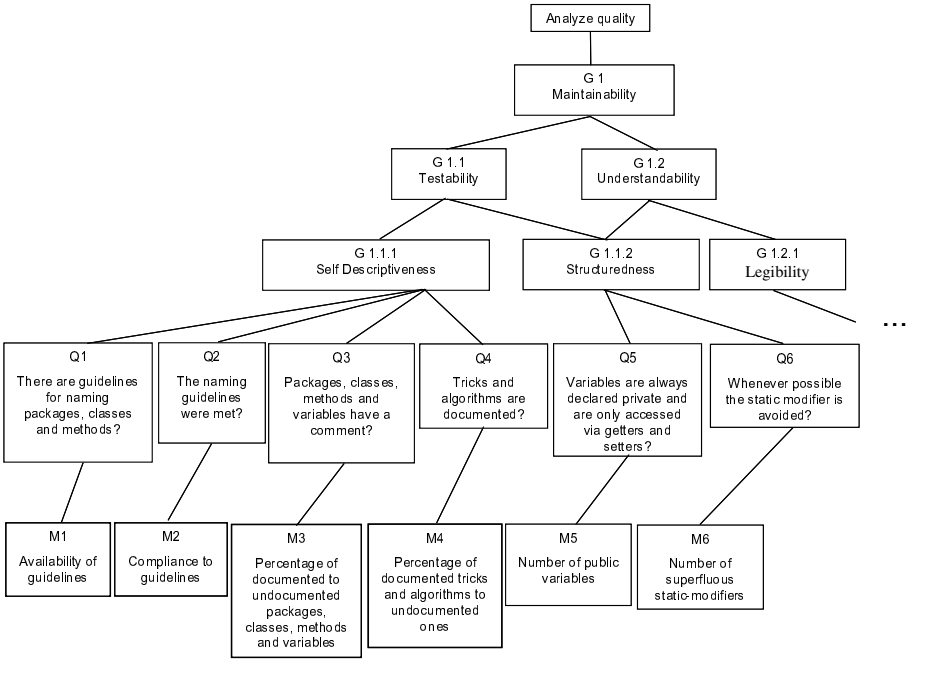


Fig. 4. Part of maintainability branch of the GQM tree

goal. For the metrics of the lowest level of the GQM tree predicates, desired values and actual measurement objects were set in order to obtain the criteria.

Overall, five new metrics could be identified through the GQM method. Six criteria were identified by both methods. Nine criteria were identified through the experienced-based approach. Nonetheless, only one of the nine criteria was not trivial, while eight criteria only asked for the existence of a special document (which was in all cases covered by at least one other criterion). Moreover, some criteria were not quality focused.

Our results indicate that an experience-based approach only covers 58% of the criteria, while a systematic goal-based approach (namely GQM) covers 92% of the necessary criteria. As a further result it can be stated that an experienced-based approach delivers a great number of trivial criteria (40%). Table 3 summarizes our results.

When defining the criteria we did not measured the time needed for each method. We estimate that the experienced-based approach took approximately 30 minutes, while the goal-based approach took at least 3 hours. Consequently, we can assume that a goal-based approach takes much more time.

Table 3. Number of identified criteria

	Experienced-based only	Experienced-based and Goal-based	Goal-based only
# of criteria	9, corrected 1	6	5
In % of the total number of 20 criteria	45%	30%	25%
In % of the corrected number of 12 criteria	8,33%	50%	41,67%

3 Experience-Based Improvement of Criteria

Even if the criteria where defined by using a systematic method we can not be sure that they are perfect. It is extremely important to foster the criteria of a fixed criteria catalogue, because they are applied to all projects. Most experiences on Quality Gates and especially on criteria are externalized during the Quality Gate review, when project results are checked against the criteria and actions must be taken. Consequently, many persons participate in the gate review. These experiences can be captured in a special experience base [6] avoiding a loss of valuable experiences. All involved persons can judge the suitability of criteria in the current project. Experiences can be captured using our workflow tool *NetQGate* [7] (see figure 5). NetQGate is realized as a web application. It provides assistance in scheduling Quality Gates and in setting the right criteria for the scheduled Quality Gates. It supports the Quality Gate review by managing all documents requested by criteria and by documenting the outcome of the review. In this way all the information needed for a successful conduction of a Quality Gate review remains in one central place.

An experience in NetQGate consists of three elements:

- An observation describing the experience and the context it is valid for.
- An emotion judging the observation from a subjective point of view.
- A conclusion containing hints how to handle the crtierion in a similar or other context.

Since 2006 NetQGate is used in our students' software projects to support their Quality Gate processes [8, 9]. Since then experiences of 16 projects were used to improve the criteria catalogue. Experiences could be used in three ways:

- Criteria were changed, deleted or combined with other criteria.
- Guidelines were attached to criteria in order to provide common interpretations.
- Rules were added indicating in which project situations criteria have to be applied.

In our software projects gatekeepers and project representatives frequently used the possibility to judge criteria. Nonetheless, a greater number of experiences was almost useless, because an observation often was absent. To judge the effectiveness of NetQGate as a part of an experience feedback cycle, we only regard experiences with a nonempty observation part. Table 4 depicts the results of criteria improvement. Overall, we can state that a considerable percentage (maximum of 18%) of the 62 criteria could be improved in at least one of the three aspects from above.

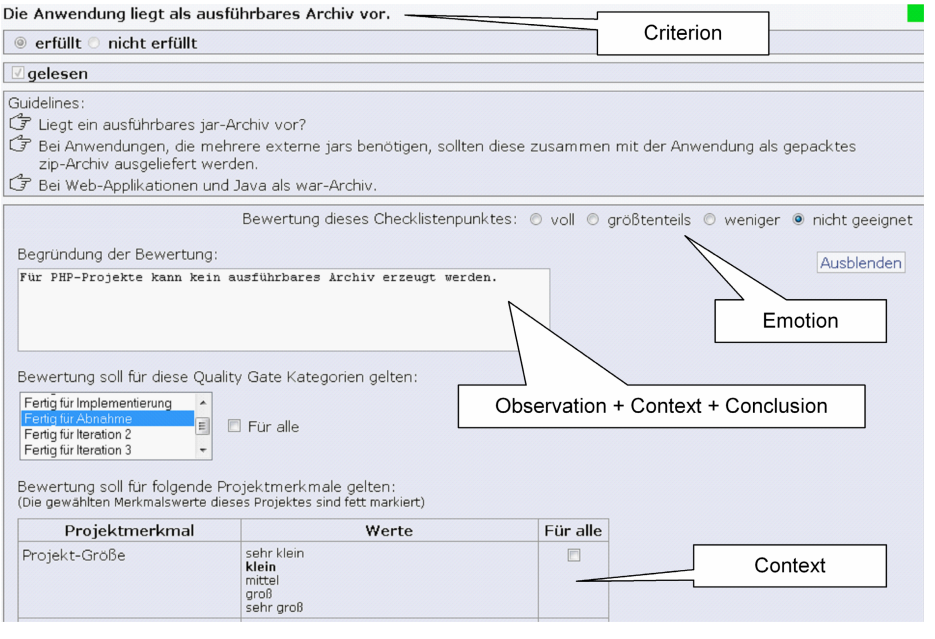


Fig. 5. Rating of a criterion in NetQGate

Table 4. Results of criteria improvement

	Changed, deleted or summarized criteria	Attached guidelines to criteria	Added rules
# of criteria	9, corrected 1	6	5
In % of the total number of 20 criteria	45%	30%	25%
In % of the corrected number of 12 criteria	8,33%	50%	41,67%

When using NetQGate as a part of an experience feedback cycle two major problems were observed:

- Without a special introduction to NetQGate, project representatives tend to misunderstand the experience capturing function. Some project representatives thought that the experience capturing function aims to rate the fulfillment of criteria.
- Project representatives tend to leave the observation part empty resulting in almost useless experiences.

Nevertheless, enough complete experiences were captured. We strongly assume that fewer experiences will be captured in future, because the criteria catalogue has become more mature.

3.1 NetQGate as Part of an Experience Feedback Cycle

A typical experience feedback cycle consists of four activities: activate, collect and prepare, store and reuse (see figure 6), which are conducted by an experience engineer.

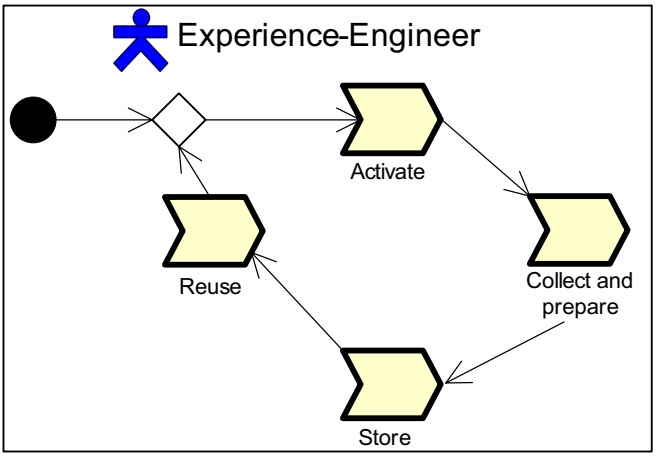


Fig. 6. A typical Experience Feedback Cycle

For each of these activities NetQGate provides assisstance in some way:

- **Activate:** NetQGate helps to activate experiences, by providing the opportunity to judge criteria with low effort and at any time within a project. Especially, experiences are made in a Quality Gate review and within the run-up of a Quality Gate, when project results are prepared according to the criteria of the Quality Gate. For convenience the project situation (= context) is automatically saved in order to better reuse the experience in future.
- **Collect and prepare:** All activated experiences are stored within a temporal store. NetQGate provides help in navigating this store in different ways. Experiences can be filtered in different ways. For example it is possible to retrieve all experiences for a special project or a special Quality Gate. Finally, after an experience has been judged by an experience engineer, it can be stored in order to be reused.
- **Store:** Prepared experiences are stored a second store. These experiences can be used in order to tailor suitable criteria for a given project situation.
- **Reuse:** NetQGate includes a simple algorithm to evaluate whether a criterion fits to a project situation or not. The input of the algorithm is a project situation, which formally describes a project e. g. in size, mission and other relevant aspects. Its output is a set of criteria for each Quality Gate.

4 Strategies for Criteria Improvement

In the previous section we had a closer look on the criteria improvement. Criteria being applied to our software projects were initially defined through an experience-based approach. Thus, we cannot assume that these criteria are perfect – even if they are defined in a systematic way. Nonetheless, we could immediately apply the criteria to projects and improve them over time. We denote this strategy as *minimum improvement strategy*. Since experiences will flow in later (after a set of projects is conducted), we can assume that the criteria are quite immature.

On the other hand it, it is possible to improve the criteria catalogue as best as possible before it is applied to a project for the first time. To achieve this aim experiences have to be carefully collected beforehand. As a result, the criteria catalogue can be used later but it is in a more mature state. Additionally, fewer experience feedback cycles are needed later. Nevertheless, this strategy leads to much more effort in the beginning. We denote this strategy as *maximum improvement strategy*.

Overall, we assume that the total effort to get a mature criteria catalogue in the end is of same size regardless which strategy is followed. Moreover, we also assume that the maturity is of the same level in the end. Figure 7 summarizes our assumptions on the evolution of maturity and effort over time. The maturity function is gained by mathematical integration of the corresponding effort function.

A software company has to set a strategy for criteria improvement. This strategy will depend on the available resources and on the knowledge of future project situations. Smaller companies with uncertain project situations will be more eager to follow the *minimum improvement strategy*, because it has lower effort peaks and better deals with changing and uncertain projects. On the other hand, large companies with stable projects will apply the *maximum improvement strategy*. In case of uncertain

projects, it is always better to follow the *minimum improvement strategy*. Smaller companies will always follow the *minimum improvement strategy*, because it has smaller resource peaks. Table 5 summarizes the selection of strategy with regard to certainty of projects and company size.

Table 5. Selection of Improvement Strategy

	Stable projects	Uncertain projects
Smaller company	*minimum improvement strategy*	*minimum improvement strategy*
Larger company	*maximum improvement strategy*	*minimum improvement strategy*

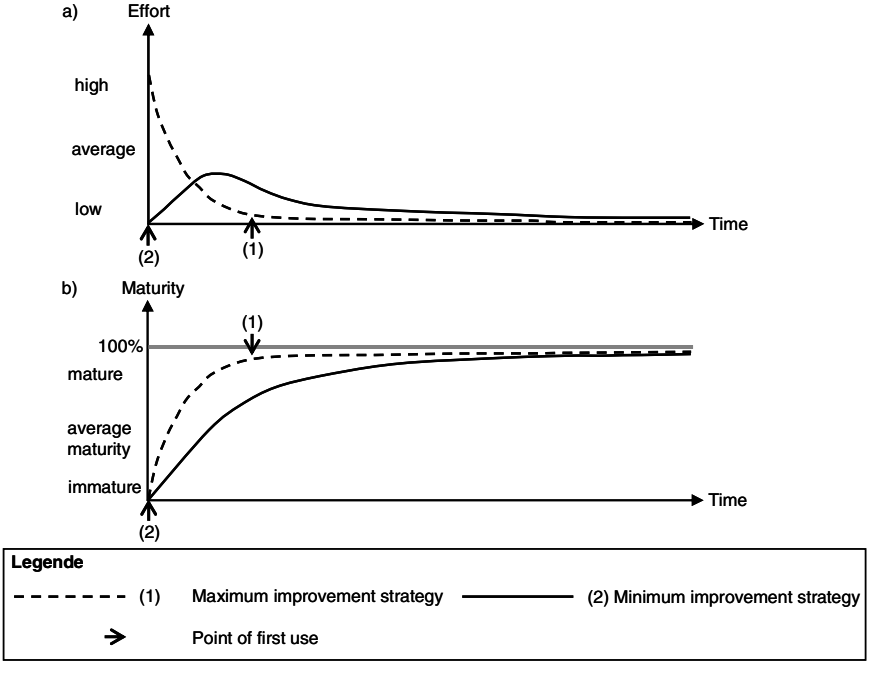


Fig. 7. Evolution of maturity and effort over time

5 Conclusion and Outlook

This paper showed how a software company can gain suitable criteria for Quality Gates. Depending on the strategy a fixed criteria catalogue has to be established and fostered. Criteria can be defined in different project phases by applying different

methods. Systematic top-down methods provide a better coverage but require more resources, while an experienced-based approach helps to save resources at the expense of goal coverage and thus quality. Overall, we can state that a systematic top-down approach delivers 92% of the criteria, while an experienced-based yields only 58% and a greater number of trivial criteria with regard to the set quality goals.

Experienced-based improvement of criteria is necessary, if a fixed criteria catalogue is used. In our setup a considerable number of criteria could be improvement in some way, when experiences are activated and captured during the Quality Gate review.

We strongly encourage systematical capturing of experiences during the Quality Gate review, because it involves a great number of roles. The Quality Gate review can be supported by a workflow tool which facilitates efficient experience capturing at the same time. We strongly believe that our workflow tool NetQGate is an excellent support here. Besides the ability to activate experiences it also helps to conduct other activities of a typical experience feedback cycle.

Nonetheless, all observations are taken from our students' software projects. According to Wohlin et al. [10] we have a low external validity here. Further evaluations in a software company's environment are necessary to gain a better external validity.

References

1. Pfeifer, T., Schmidt, R.: Das Quality-Gate-Konzept: Entwicklungsprojekte softwareintensiver Systeme verlässlich planen. Industrie Management 19(5), 21–24 (2003)
2. Hawlitzky, N.: Integriertes Qualitätscontrolling von Unternehmensprozessen - Gestaltung eines Quality Gate-Konzeptes. In: Wildemann, H. (ed.) TCW Wissenschaft und Praxis. TCW Transfer-Centrum, München (2002)
3. Solingen, R.v., Berghout, E. (eds.): The Goal/Question/Metric Method - A Practical Guide for Quality Improvement of Software Development. McGraw-Hill, New York (1999)
4. Schneider, K.: Abenteuer Softwarequalität, p. 212. dpunkt verlag, Heidelberg (2007)
5. Knauss, E.: Einsatz computergestützter Kritiken für Anforderungen. GI Softwaretechnik-Trends 27(1), 27–28 (2007)
6. Basili, V., Caldiera, G., Rombach, H.D.: Experience factory. In: Marciniak, J.J. (ed.) Encyclopedia of Software Engineering, pp. 469–476. John Wiley & Sons, New York (1994)
7. Flohr, T.: NetQGate - Tool Support for Quality Gate Processes. In: 9th International Conference on Quality Engineering in Software Technology (CONQUEST). dpunkt verlag, Berlin (2006)
8. Lübke, D., Flohr, T.: Simulated Software Project Driven by Quality Gates. In: Electronics World, 2006, vol. 1840, pp. 38–42 (2006)
9. Lübke, D., Flohr, T., Schneider, K.: Serious Insights through Playful Software-Projects. In: Dingsøyr, T. (ed.) EuroSPI 2004. LNCS, vol. 3281, pp. 57–68. Springer, Heidelberg (2004)
10. Wohlin, C., et al.: Experimentation In Software Engineering: An Introduction. Kluwer Academic Publishers, Dordrecht (2000)

An Empirical Study of Product Measurement in a Standardized Requirement Definition Process with 28 Japanese Government Software Projects

Yoshiki Mitani[1,2], Tomoko Matsumura[2], Mike Barker[2],
Seishiro Tsuruho[1,3], Katsuro Inoue[4], and Ken-Ichi Matsumoto[2]

[1] Information Technology Promotion Agency, Japan (IPA)
[2] Nara Institute of Science and Technology (NAIST)
[3] Kochi University of Technology,
[4] Osaka University
{ymitani,tomoko-m,matumoto}@is.naist.jp, mbarker@MIT.EDU,
tsuruho@ipa.go.jp, inoue@ist.osaka-u.ac.jp

Abstract. This paper presents an empirical study in the requirement definition process using standardized process and product formats. The results indicate that the measurement of product quantities is useful for project management and evaluation. Previously empirical research of the requirement definition process was difficult, but it became easier in the field of governmental business system optimization because the Japanese government adopted standards. In this paper, the authors evaluate and compare results of 24 projects that gathered measurements and prove that the results of the previous authors' study in one project can be generalized. In addition, the paper presents a study about the possibility of project evaluation using such standardized product measurements.

Keywords: Requirements Definition Phase Measurement.

1 Introduction

The authors have previously verified the usefulness of in-process project measurement and feedback to project management in development [1][2]. In the software develop-ment process, the authors had focused their interest on design, coding, and testing phases.

The previous empirical study combined measurements and analysis to provide feedback to the project management. This produced positive effects for the project. Based on the success of this research, the authors wanted to expand in-process measurement to the "requirements definition" phase and to develop a complete lifecycle measurement method for processes and products across the full development process.

2 Previous Measurement Experiment

2.1 Motivation

Most previous research in the requirement definition process focused on the definition activity itself, rarely focusing on process standardization and measurement. In the

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 257–270, 2008.

SWEBOK, there is some description about Software Engineering Management, but it is not developed into measurement of this process [3]. Although the draft of Chapter 12 which is going to add a measurement view to SWEBOK has some description about Measurement by SLC phase, SW Requirement [4], it focuses on measurement of just the number of requirements and makes no reference to process measurement.

The research called The Measurement Dashboard focused on in-process measurement of the software process [5]. In this research various measurement targets were included, such as milestones, earned value, expense and number of requirements. But in the requirement definition process, while it included management of requirements such as change control of requirements in total software process, there was no interest in in-process measurement of product.

Empirical study and measurements of the requirements definition phase has been difficult because there are various methods and tools used and there is relatively little standardization. However, the use of the Enterprise Architecture method in Japan makes it easier to measure this phase.

The Enterprise Architecture is a total methodology for enterprise information system development, arranging the organizational business and information systems to provide overall optimizations. It is based on Zachman's framework [6] and constructed from various proposed design and management methods.

Practically, the EA method consists of three phases, the AsIs, ToBe, and policy arrangement phases. The AsIs phase describes the present state of the target system. The ToBe phase designs the future ideal system. The policy arrangement phase, between the AsIs and ToBe phases, makes decisions about policies for optimizing the business and systems. A special feature of the EA method is that it defines many standardized hierarchical diagrams for the AsIs and the ToBe phases to increase the mutual understanding between target system stakeholders.

From the viewpoint of project measurement, it is easier to expand project measurement methods from the development phase into the requirements definition phase because the EA method has standardized process and product formats. The in-process measurement can use the amount of descriptions and the transitions in diagrams in the AsIs and ToBe phases, such as the total number of diagrams, along with additions, eliminations, and modifications of diagram elements.

2.2 Background

In 2003, the Japanese government provided an Enterprise Architecture (EA) guideline for the requirements definition phase to optimize government business [7]. This guideline provides three phases, AsIs, Optimize, and ToBe, as the process architecture.

In the AsIs and ToBe phases, the requirements definition uses four kinds of diagrams: The Diamond Mandala Matrix (DMM), Data Flow Diagram (DFD), Work Flow Architecture (WFA), and Entity Relationship Diagram (ERD), as examples are illustrated in Fig. 1 to Fig. 4. The characteristic of this method is to describe a large quantity of these diagrams.

Previously the authors had an opportunity to measure one requirement definition process of a Japanese governmental project which indicated that such measurement

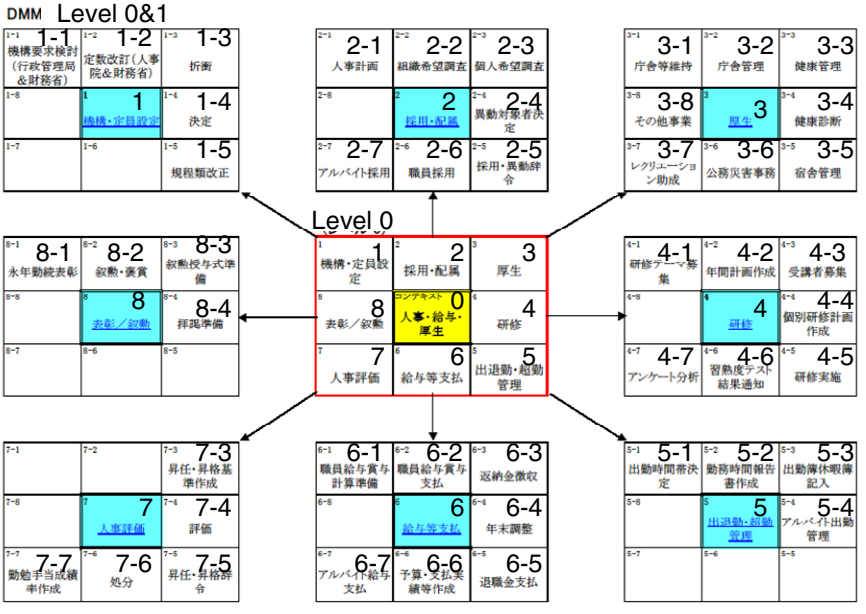


Fig. 1. Diamond Mandala Matrix (DMM)

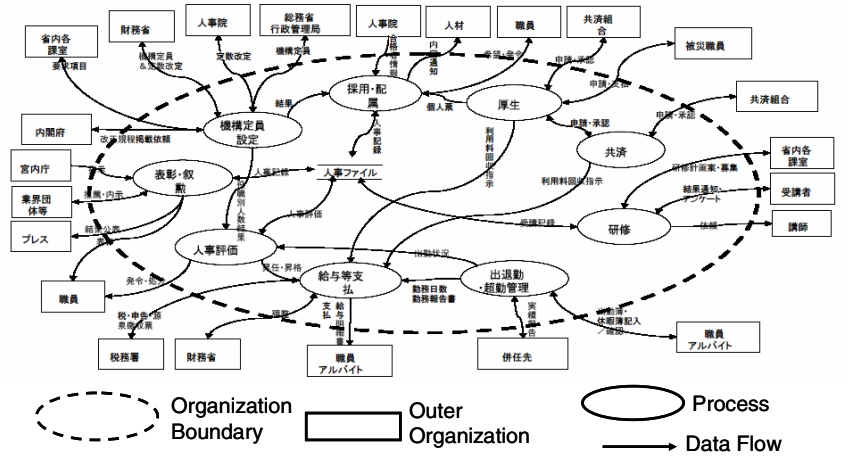


Fig. 2. Data Flow Diagram (DFD)

was useful for project management [8][9]. After that the Japanese government introduced to the public results of over 40 projects' requirement definition to get public comments. In this research authors evaluate and compare previous data with the new publicly opened data of 24 systems and investigate the possibility of project evaluation.

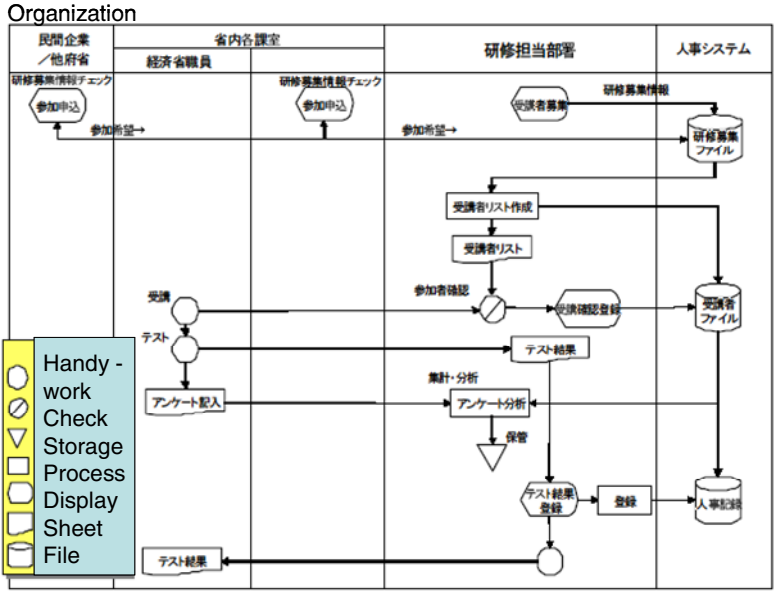


Fig. 3. Work Flow Architecture (WFA)

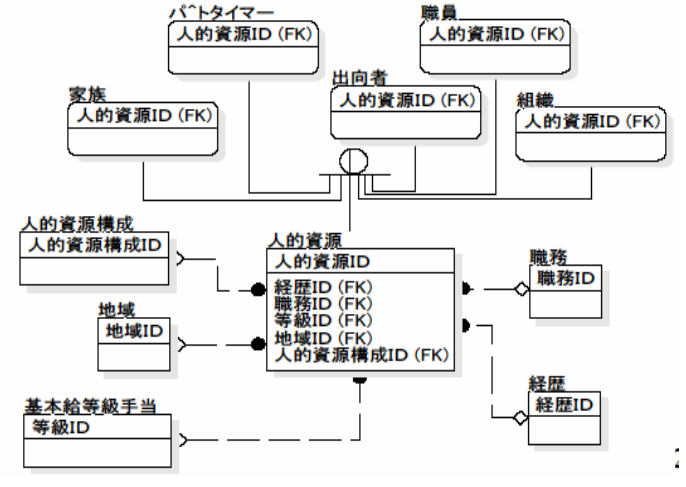


Fig. 4. Entity Relationship Diagram (ERD)

2.3 Measurement of Requirement Definition Products

As reported in previous papers [8][9], in 13 weeks, the description of the AsIs diagrams was completed for three businesses and another one was completed in the end of May 2007. ToBe diagram description for three businesses started January 2007 and finished after 11 weeks work.

During this process, the following measurements and graphical visualizations were made:

1) The amount of diagram description and transitions measured in number of sheets, both total amounts and for each business
2) The number of diagram elements and diagram connector elements in the diagrams and their transitions, measured as total amounts and for each business
3) Changes in the numbers of diagram files through addition, elimination, and modification, both total and for each business
4) For each diagram in each file, measure the addition, elimination, and modifications of elements by counting text strings on the diagram elements.
5) Weekly described amount of diagram transitions by number of sheets, diagram elements, and connector elements, both total and for each business.

Fig. 5 to Fig. 11 shows examples of these measurement results in graphical form. The following trends are directly read from those graphs.

Fig. 5 shows changes in the described diagram elements with the cumulative stack of each business during the 24 weeks. In total, about 730 sheets, 34,000 elements were described. This graph shows not only the total amount project proceeding process for each business but also description documents amount, working start timing and finished stable situation.

Fig. 6 shows the number of AsIs described elements for business B. This study illustrates the data for each business. Fig. 7 and Fig. 8 show the file number status of AsIs and ToBe phase of Business B. It shows that the AsIs phase smoothly progressed to stable and the ToBe phase also rather rapidly progressed. Fig. 9 shows all changes

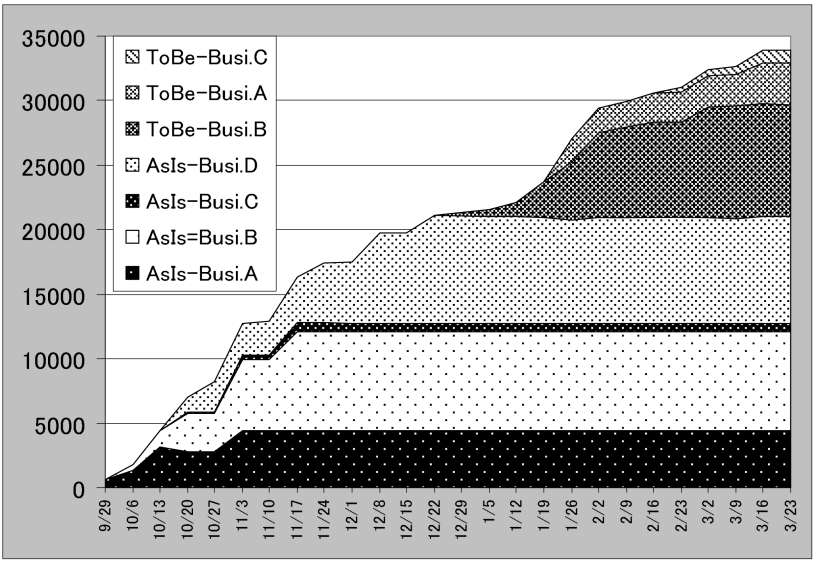


Fig. 5. Diagram Elements Stack of all Business

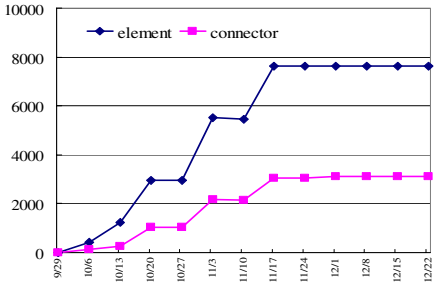


Fig. 6. Diagram Elements of Business B (AsIs)

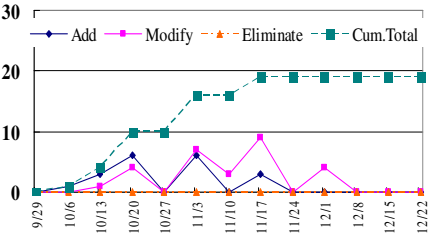


Fig. 7. File Number Transition of Business B (AsIs)

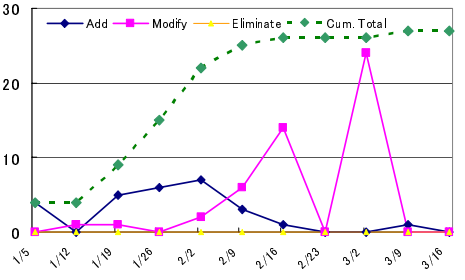


Fig. 8. File Number Transition of Business B (ToBe)

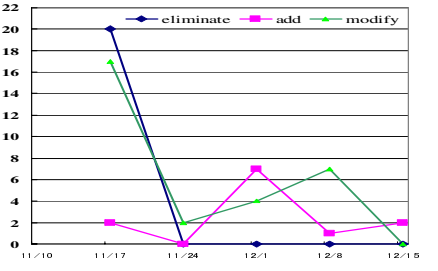


Fig. 9. Diagram Modification in one file Example (8 sheets)

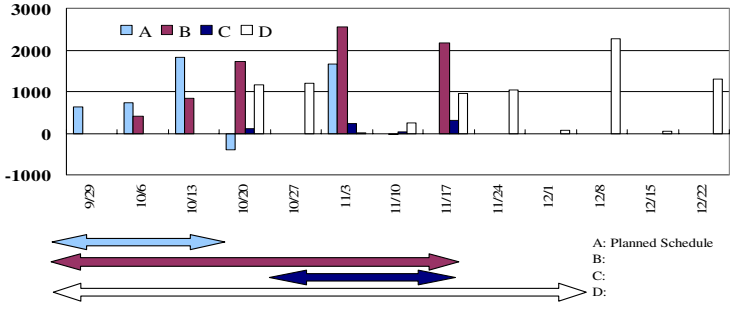


Fig. 10. Weekly Addition of Diagram Elements (AsIs)

of one diagram file consisting of eight diagrams included in 6 weeks. In this figure, the trend appears to show that this area's AsIs description work is gradually stabilizing. Fig. 10 showed changes in the amount of description on a weekly basis in AsIs phase case. From this graph, it is clear that the description process for the four businesses were executed shifted a few weeks. Fig. 11 shows the total changes on a weekly basis. The amount of work performed can be estimated from that figure.

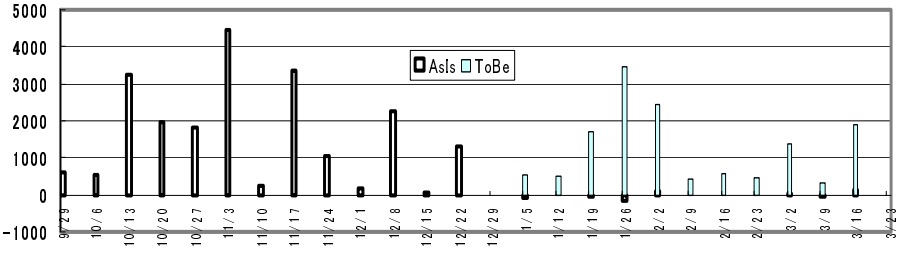


Fig. 11. Total Weekly Addition of Diagram Elements (AsIs and ToBe)

2.4 Comparison with Official Progress Report

Based on the governmental EA guideline, the target project is managed with EVM and WBS methods. The official progress report is based on declarations by the participants. Fig. 12 is an example of the EVM report. This report shows the consumed human resources, but it is not clear about the situation of the outcome amounts. Fig. 13 shows the WBS declaration level progress report visualized by authors. In the WBS method, the work in progress is reported through detailed activities. However, the granularity of this report is very rough. There is no information about the amount of outcome produced. The WBS based report is based on declared progress estimation criteria. This reporting is also limited by self declaration and human intervention. For example, in some case progress raised rapidly to 80% but after that it remained stable for a long time, or in the other case, in the EVM chart, during a long period progress was delayed but when the deadline was coming it progressed rapidly and finished on time.

Compared to these official reports based on self-declarations, the product measurement method tried in this study presents detailed information with high granularity based on real amounts of outcome products. This information is based on the raw data of the production, so human intervention does not affect it. For example, Fig. 10 includes the declared schedule of work. From this graph, gaps between the declared schedule and the real work progress based on actual product information are clearly visible.

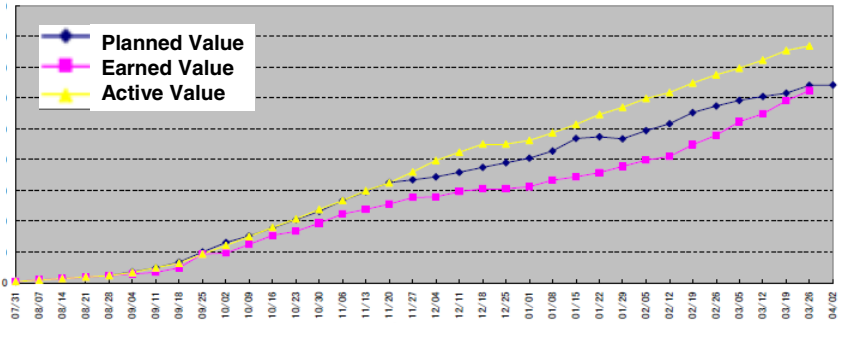


Fig. 12. EVM Report

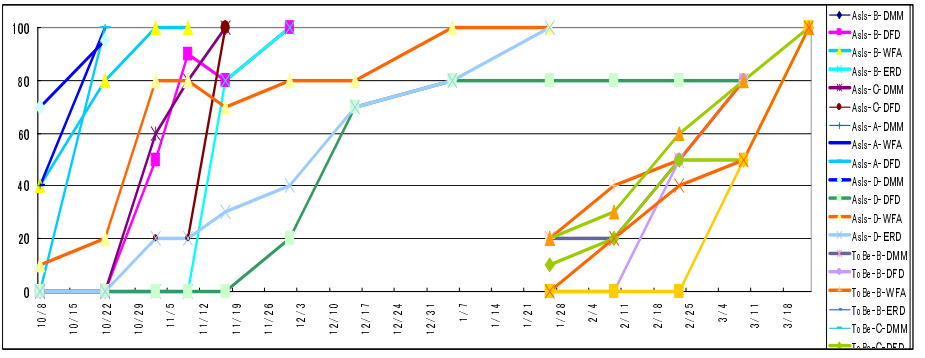


Fig. 13. Declaration Level Progress Report (AsIs & ToBe) (%)

2.5 Study for New Software Metrics Possibility

The target system for measurement in this study is not very large, but during 24 weeks, this work included over 700 diagram sheets with over 30,000 diagram elements. Considering analogies between measurement targets such as diagram sheets or numbers of diagram elements and program modules or source lines of code (SLOC), we can propose developing new software metrics for the requirements definition phase. These new metrics are expected to contribute to increasing the productivity and quality of software development processes in the same way as other existing software metrics. For example, it is likely that 30,000 diagram elements are analogous to 30 Ksteps of high-level programming language code, the number of diagram sheets can be compared to the number of program modules, and the number of diagram file as corresponds to the number of program files.

For example, Fig. 14 illustrates diagram element numbers per working effort. As other metrics, working effort per sheet, working effort per diagram element, numbers of sheets per working effort were considerable. Working effort can be converted into cost.

This trial is based on one project case study but comparing seven business works we can see differences in working density. For example, productivity depends on each business. Both businesses A and C had fewer products but their productivity shows different trends. In the case of business A, it was easy to understand business process, so it showed high productivity but in the case of business C, the business process was highly complicated, so low productivity was shown. This trend is not same as the

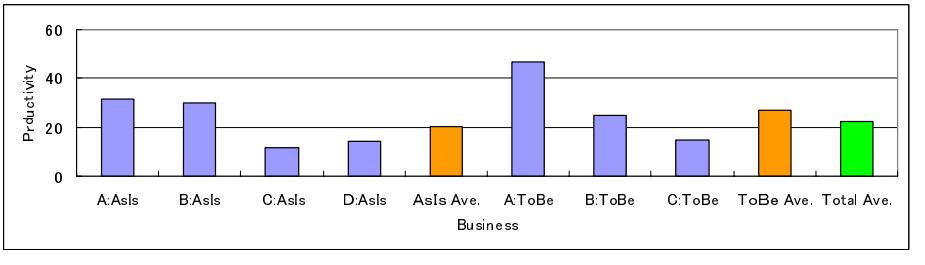


Fig. 14. Diagram Elements per effort

general trend in the development phase measured by SLOC and Function Point (FP). In the development phase, generally larger development has lower productivity.

3 Evaluation of the New Publicly Opened Data

The newly introduced diagrams were in PDF form. They were limited to final result documents only with no process data. Given these limitations, the authors selected 24 systems from about 40 systems, which included a large quantity of diagrams such as DMM, DFD, WFA, ERD, and compared them with the previously measured four IPA subsystem data.

Authors compared sheet numbers in AsIs and ToBe phase, and sorted them with total sheet numbers. Fig. 15 shows example of previous measurement which shows changes in the described diagram sheet numbers with the cumulative stack of each business during 24 weeks. In total, about 730 sheets and 34,000 diagram elements were described as previously reported.

Then authors got Fig. 16 histogram.

In Fig. 16, position of IPA four subsystems are 9th, 14th, 15th and 25th, which means that the previous measurement target were average systems in high-end, middle and low scale systems, not outliers. This result suggests that the results of the previous measurement experiment have generality in the requirement definition phase under the same guidelines.

That the results of the previous measurement experiment in only one project have generality indicates that the measurement of output diagrams and transitions provides a useful way to characterize project progress. It suggests that the measured data in the requirement phase can be used in a similar way to SLOC in the programming phase, creating a new software metrics.

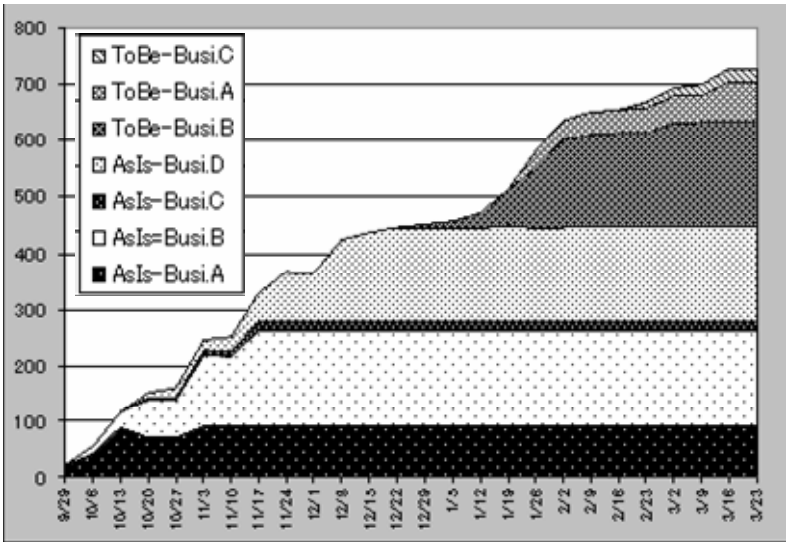
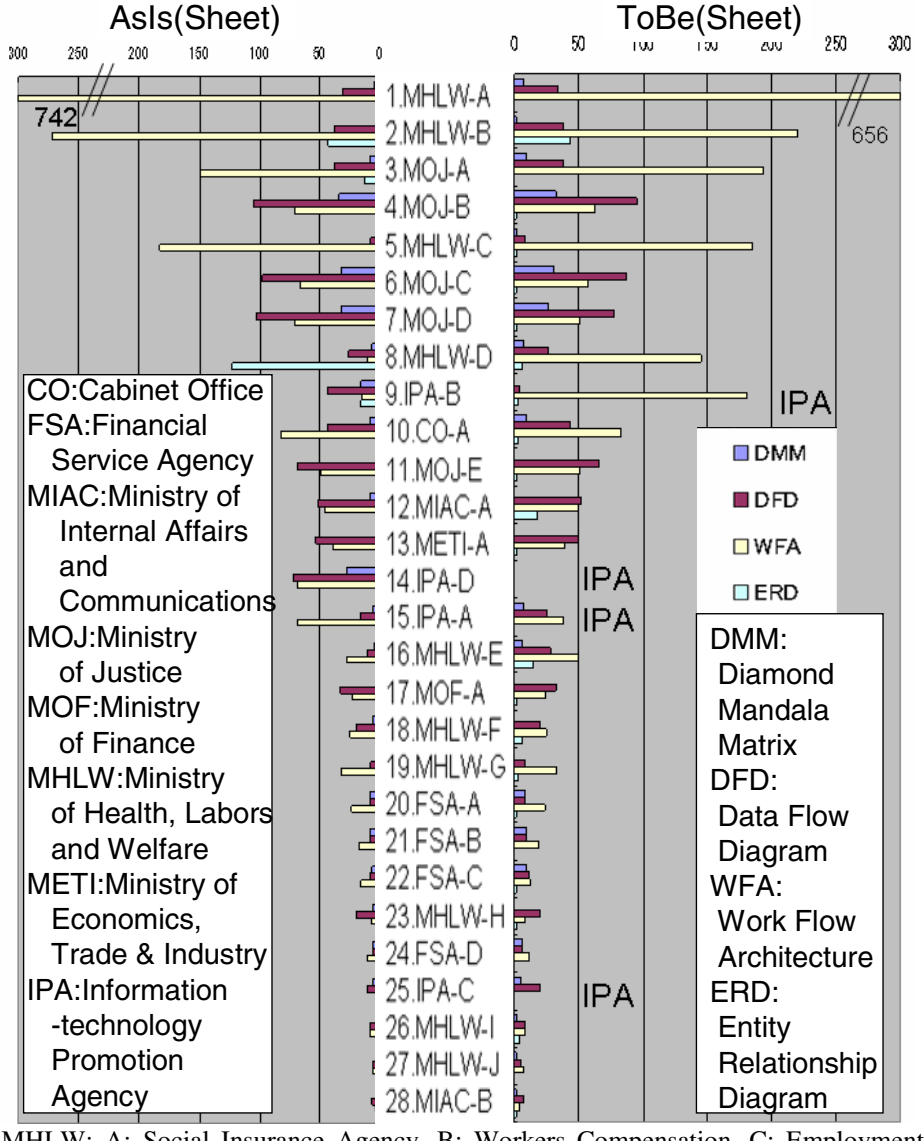


Fig. 15. Diagram Sheets Stack of all Business in the Pervious Measurement



MHLW: A; Social Insurance Agency, B: Workers Compensation, C: Employment Security, D: Labor Insurance, E: National Cancer Center, F: Food Safety, G: Industrial Safety and Health, H: Quarantine, I: National Peace Memorial Hall, J: Equal Employment, MOJ: A; Immigration, B: Registration Information, C: Map Information, D: Correction Bureau, E: Rehabilitation Bureau, CO: A; Economic and Fiscal Policy, MIAC: A: Statistics Bureau, B: Information and Communication policy, METI: A; Japan Patent Office, MOF: A; Mutual Assistance, FSA:, A; Supervisory, B: Inspection, C: Electronic Disclosure for Investors' Network, D: Securities and Exchange Surveillance,

Fig. 16. Diagram sheet number of AsIs and ToBe phase on 28 government systems

Fig. 16 suggests that the measurement of product quantity of standardized requirement definition phase is useful for project evaluation. Detail evaluation needs farther project context information, but from opened data, for example, following characteristics became clear.

From the total amount of described diagrams, the total scale of developing system can be predicted. For example, as shown in Fig. 17, the scale of the Ministry of Health, Labors and Welfare (MHLW), Social Insurance System (1) is projected.

In some systems, the description amounts of AsIs and ToBe are quite different. For example, as shown in Fig. 18, comparing to MHLW, Labor Insurance System (8), , in the IPA Business B System (9), the description amounts of ToBe are remarkably larger than AsIs. In IPA, business D System (14), ToBe diagrams were not described.

In some systems, the ratio of the kinds of the diagram is quite different between AsIs and Tobe. For example, in MHLW, Labor Insurance System (8) a lot of ERD were described in AsIs phase but in ToBe phase more WFA were described.(Fig. 18)

The ratio of the kinds of the diagram depends on each system, for example, as shown in Fig. 19 and Fig. 20, in some systems there are a lot of DFD and in another system there are a lot of WFA. In large scale systems there is a tendency to describe large amount of WFA. In some systems there are no ERD.

Generally, the requirement definition process was not so visible until now, but in the standardized process it becomes very visible by measuring its products.

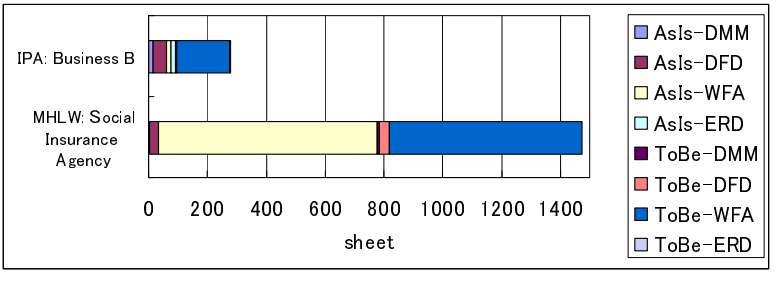


Fig. 17. Comparison example of total description

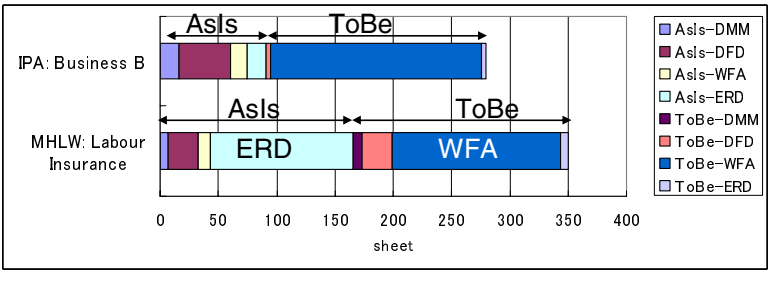


Fig. 18. Comparison example of AsIs / ToBe description ratio

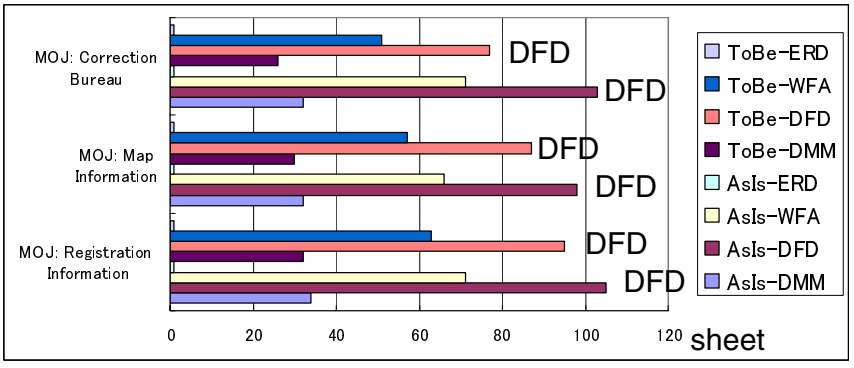


Fig. 19. Example of a lot of DFD description

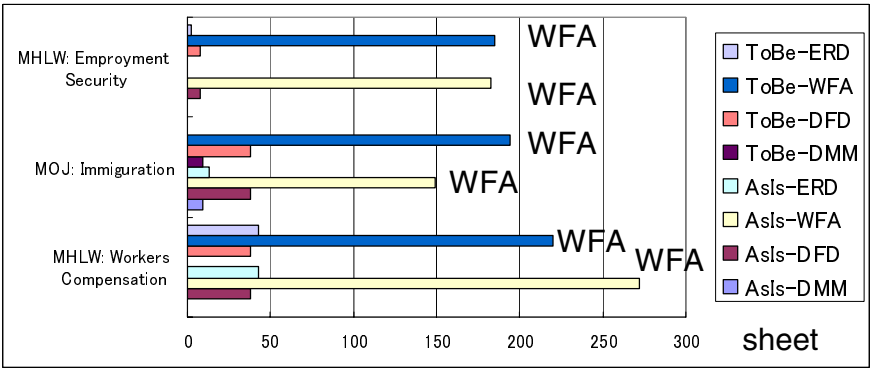


Fig. 20. Example of a lot of WFA description

4 Consideration about a New Measurement Opportunity

In software engineering research, the following two issues complicate the results:

1) Confidentiality of software project.
2) Independence of software project.

The authors' measurement of products of requirement definition from various governmental development projects suggests a possible way to conquer these issues.

The government finds it necessary to get public comments on requirements definition for their own system development.

The government also has strong motivation to standardize their own system development process from productivity viewpoint.

In addition, there are many development demands in the governmental sector. In Japan, it occupies 10% of domestic system development market.

This situation gives software engineering research wonderful materials to do research on process measurement in requirement definition that was difficult in former times. The products of the requirement definition process introduced to the public by the government are useful bases for approaches that conquer the confidentiality and independence of software project in software engineering research.

5 Conclusion

The results of a previously presented empirical study of a standardized requirement definition phase measurement in one system (included four sub-systems) was compared and evaluated in relation to data from newly opened governmental products in over 24 projects, and its generality was verified. In the case of standardized requirement definition phase, in-process measurement of its product is useful for project management and it brings us new software metrics comparable to the SLOC or module numbers in the programming phase. Those measurement brought visibility into requirement definition process which was previously invisible.

Acknowledgments. This work is supported by IPA/SEC, METI and the MEXT of Japan. We thank researchers in the SEC and StagE project who kindly support our project.

References

1. Mitani, Y., Kikuchi, N., Matsumura, T., Ohsugi, N., Monden, A., Higo, Y., Inoue, K., Barker, M., Matsumoto, K.-i.: A Proposed Method for Building a Database of Project Measurements and Applying it Using Collaborative Filtering. In: Proceedings of 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2006), Rio de Janeiro, Brazil, October, vol. 2, pp. 15–17 (2006) Short papers
2. Mitani, Y., Kikuchi, N., Matsumura, T., Ohsugi, N., Monden, A., Higo, Y., Inoue, K., Barker, M., Matsumoto, K.-i.: A Proposal for Analysis and Prediction for Software Projects using Collaborative Filtering. In: Process Measurements and a Benchmarks Database. MENSURA 2006, International Conference on Software Process and Product Measurement, Cadiz, Spain, pp. 98–107 (November 2006)
3. IEEE: The Software Engineering Body of Knowledge (SWEBOK) (2004)
4. Chapter 12 Software Measurement (SWEBOK), Draft May 1 (2007) http://mensura2007.uib.es/documents/SwMeasBok_Ch12_vMay192007.pdf
5. Ebert, C., Dumke, R.: Software Measurement. 8.3 Managements for Project Control, 199–228 (2007)
6. Zachman, J.A.: A Framework for Information Systems Architecture. IBM Systems Journal 26(3) (1987)
7. METI: Enterprise Architecture, http://www.meti.go.jp/english/information/data/IT-policy/ea.htm

8. Mitani, Y., Matsumura, T., Barker, M., Tsuruho, S., Inoue, K., Matsumoto, K.-I.: Proposal of a Complete Life Cycle In-Process Measurement Model Based on Evaluation of an In-Process Measurement Experiment Using a Standardized Requirement Definition Process. In: Proceedings of International Symposium on Empirical Software Engineering and Metrics 2007 (ESEM 2007), Madrid, Spain, September 2007, pp. 11–20 (2007)
9. Mitani, Y., Matsumura, T., Barker, M., Tsuruho, S., Inoue, K., Matsumoto, K.-I.: An Empirical Study of Requirement Definition Process Management and Metrics based on an In-process Measurement Experiment of Standardized Requirement Definition Phase. In: Proceedings of International Conference on Software Process and Product Measurement (IWSM-Mensura 2007), Palma, Spain, November, pp. 121–131 (2007)

Measuring 75 Million Lines of Code

Harry M. Sneed

ANECON GmbH
A1091 Vienna, Austria
Institut für Wirtschaftsinformatik, Universität Regensburg
Harry.Sneed@T-Online.de

Abstract. The following paper describes a measurement project to measure and evaluate the software application systems of a financial services provider. Due to several mergers the cooperation had accumulated over the years more than 75 million lines of code in several different programming languages. The goal of the project was to determine the size, complexity and quality of the different systems and to evaluate their potential reuse. Not only the program source, but also the database schemas, the JCL procedures and the user interface maps had to be analyzed. For this purpose a metric database was established. In the measurement project three related tools were used. The tool *SoftAudit* was deployed to measure the code. The tool *SoftEval* was used to aggregate the measurement data in a metric database and to evaluate it. The tool *SoftCalc* was used to calculate the costs of various strategic alternatives. The paper focuses on the problems and solutions associated with such a massive measurement effort of large code bases.

Keywords: Code measurement, size, complexity and quality metrics, metric database, metric evaluation, ISO-9126.

1 Introduction

The financial service provider in question has gone through several mergers in the past years. Each merger has brought in additional IT systems, some of which have been integrated but most of which have remained independent. The preservation of the former independent business units is necessary in order to preserve the previous corporate identities and too ensure the continuity of the financial service for the customers. On the other hand, it has always been a goal to merge the internal administrations of the previously independent organizations in order to reduce costs. This includes the IT departments. For the IT activities, a separate daughter company was founded as an internal software house, whose mission it is to provide the necessary information technology support to the line organizations.

The merger of IT activities seemed at first glance to be a straight forward and simple solution. The former independent companies were all IBM customers. Their software systems ran on an IBM mainframe, their users were equipped with IBM terminals or PC's and all of the data was stored in an IBM database. Thus, from a hardware point of view, everything appeared to be in a common blue color.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 271–286, 2008.

However, when looking at the software, a great deal of diversity can be discovered. The previously independent IT organizations used different languages in different architectures. Two of the merged companies had always been staunch PL/I users. They relied on NPT - normalized program technique – generators to generate their program control structures. In the meantime they had started moving their data from hierarchical to relational data bases. The software architecture mirrors the structure of the user department. For every business process there is a corresponding software system in the classical silo type architecture.

The other merged organization had originally been an Assembler/COBOL shop. In between they had developed some new applications with the IBM ADCYCLE 4GL language CSP. They had also gone through a short PL/I phase before deciding to return to COBOL. In the meantime they, too, have been moving their data from IMS to DB2. Their software is structured in architectural layers with a data access level, a common service level, a business application level and a user interface level. This IT can be viewed as a typical layered software architecture with specialized components.

The main organization has always been a COBOL user but has its own unique architecture built around the IMS-DB/DC core environment. This architecture allows for a user-driven development. The user departments write the business rules in a formalized German language pseudo code from which control files are generated which are interpreted at run time by the COBOL modules running under an IMS teleprocessing monitor with Assembler supplements. There are more than 20 different types of syntax for describing the various artifact types – map descriptions, data views, business rules, control flow, etc. The report generating programs are written in the 4GL language *Easytrieve*. Thus, the user is able to feed his changes and new requirements directly into the system via the pseudo code language which allows him to define the maps or user interfaces. All of this sounds very futuristic with domain specific languages, code generators and ready made program frameworks, but it is very dependent on the underlying database and data communication system IMS and the user interface technology of the 1970's. Furthermore, it entails a mix of languages in almost every subsystem single online program.

In summary, from a software point of view, the former independent IT departments have taken quite different approaches to solving their application problems so that their systems reflect all of the variations of the IBM world from Assembler to Java. Even in their newer client/server applications differences exist. The main organization has developed its external services in C++ while the internal intranet applications are in Java. The other organizations are also moving in the direction of Java. The relative autonomy of the different daughter companies, the lack of development standards and the many technology waves which have swept over the IT departments in the past years have all contributed to a highly heterogeneous software world.

2 Goals of the Measurement Project

The goals of the measurement project were threefold:

- To measure the source
- To evaluate the measurements
- To calculate the strategic alternatives

The first goal was to measure the size, complexity and quality of the code base. The code base consisted not only of program source code, but also of database schemas, map definitions and job control procedures. It was intended to measure each and every source member and to aggregate the metrics at the system, product and departmental level.

The second goal was to set up and populate a metric database with which the metrics could be evaluated and various statistical reports produced including rankings, charts, graphs and dashboard gauges. This was necessary in order to access the software product quality in accordance with the criteria prescribed in the ISO Standard.

The third goal was to calculate the costs of alternative reuse strategies, namely new development, renovation, migration and integration as well as the annual costs of continual maintenance. These estimates were to be based on the measurements of the source code. In another parallel project the costs of further development were being estimated on the basis of the requirements. The first step in the measurement process was then to select, which measurements to take as proposed by the ISO Standard 9126 on software product evaluation [1].

3 Selecting the Metrics

The first step in a measurement project is to select the metrics to be used. Software systems are complex constructs with many facets, so it is not at all obvious how they should be measured. The hundreds of measurements proposed in the literature, Zuse has identified more than 300 [2], are all indicators of some kind or another, but each measurement is targeted to one particular facet of the software. The McCabe metric is focused on the control flow graph [3]. The Halstead metric is focused on the use of the programming language [4]. The Chidamer/Kemmerer metrics are focused on the static structure of an object-oriented system [5]. The function point metric is focused on the interactions between a system and its environment [6]. Even compound metrics like Oman's maintainability index are a set of individual metrics dealing with one aspect of the software, here the changeability of the code [7].

Every proponent of a particular software measure is concentrating on one or more features of the software system in the belief that these features are an indicator of the software as a whole. The problem with this, is that there is an endless number of features which could be considered and counted and, that as of to date, no one has ever been able to demonstrate that one feature or set of features is really representative of the others. Every year new metrics emerge, each claiming to be a vital indicator of size, complexity or quality of software. Unfortunately, there is no one universal measurement of software, neither for the size, nor for the complexity nor for the quality, since every measurement is measuring something else.

Furthermore, little help can be provided by the standards community. There is, unfortunately, no one universal standard. On the contrary, there are many, partly contradictory standards. The ANSI-IEEE offers:

- IEEE Std. 982: Standard Dictionary of Measures to produce reliable Software
- IEEE Std. 1045: Standard for Software Productivity Metrics

- IEEE Std. 1061: Standard for a Software Quality Metrics Methodology
- IEEE Std. 14143: Standard for Software Productivity Metrics [8].

The ISO/IEC offers the following standards on metrics:

- ISO/IEC 9126: Software Product Evaluation
- ISO/IEC 15939: Software Measurement Process [9]

Besides these standards, dealing solely with measurement, there are several other standards proposing metrics to measure the degree of compliance, standards such as:

- IEEE Std. 828: software Configuration Management
- IEEE Std. 829: Software Test Documentation
- IEEE Std. 830: Software Requirements Specification
- IEEE Std. 1016: Software Design Descriptions
- IEEE Std. 1219: Software Maintenance Processes
- ISO/IEC 12207: Software Life Cycle Processes [10].

Consequently, it is not a question of too few metrics, but more a question of too many. From all those metrics put forth in the various standards, it is not at all clear which should be used for what purpose.

The author has been dealing with this question since his involvement in the METKIT project, an EU sponsored research project to develop and propagate software metrics in the European community. This project was launched in 1990 and lasted until 1993. It resulted in a set of courses and tools intended for introducing metrics into industry. It was within this project that Norman Fenton categorized metrics into product, process and resource metrics [11] and it was in this project that the author divided the product metrics into the three dimensions of size, complexity and quality [12].

3.1 Size Metrics

To measure any one dimension of a software system, many metrics are required, one for each view of that dimension. For measuring the size of a system, one can view the code mass and count lines or statements, whereby one should be aware that they are not identical. One can also view the design model and count the number of design elements such as classes, attributes and operations as is done in the object-point method. Another view is that of the data model – the entity relationship diagrams – where it is possible to count entities, attributes, views and relationships. The results are data-points. A fourth view is that of the data flow into and out of the system. One can count inputs, outputs, files and database accesses as was done by Albrecht to come up with so called function-points. The latest trend is to count use cases with their steps and interactions to come up with use-case points. Finally, one can also count the test cases required to cover all of the code branches in white box testing or all of the requirements in black box testing. There are an unlimited number of characteristics, which could be counted to determine the size of a software system, so that leaves the counter with the burden of decision.

At the time of the METKIT project, the author decided to measure size in terms of code lines, statements, data-points and function-points. Later object-points, use-case points and test-points were introduced. Object-points can be easily extracted from the code – there one is counting classes, attributes and methods – provided the code is object-oriented [13]. They can not be derived from procedural code. Use case points cannot be taken from the code at all nor can test-points. Use case points can only be derived from the requirement specification and test-points from the test specification. Thus, in analyzing procedural source code only lines of code, statements, function-points and data-points can be counted. All four of these size measurements were used here in this project to compare the sizes of the systems and to make the necessary calculations.

It becomes obvious here, that no matter how many different characteristics of software are counted there will always be some other potentially vital characteristics which are not counted. So the selection of characteristics to count is always an arbitrary decision on the part of the counter, based on some assumptions he has about the nature of that software and what size unit could correlate with whatever goal he is striving to attain, e.g. to estimate effort or to predict defect rates.

3.2 Complexity Metrics

The second dimension of software is that of complexity. Whereas size can be defined as the number of elements in a given set, complexity is defined as the relation of the number of elements to the number of relationships between those elements

$$complexity = \frac{elements}{relationships}$$

assuming that each element has at least one relationship. Just as there are many sizes, there are also many complexities. When building his first measurement tool in the METKIT project – PC-MESS, the author was concerned about selecting the major complexities of a software system as seen at that time [14]. On the other hand, there should not be too many different complexity measurements as that would only confuse the user. So the number was restricted to 8. The eight complexity metrics selected were:

- data complexity
- data flow complexity
- data access complexity
- interface complexity
- control flow complexity
- decisional complexity
- branching complexity
- language complexity.

In this way there were three data complexity metrics, three procedural complexity metrics, a component interaction metric and a language usage metric. These eight complexity metrics are still used to measure procedural code as was the case in this project.

The data complexity metric of Chapin measures the relation between arguments, results and predicates, i.e. control data [15]. The data flow complexity metric of Elshof measures the relation between the number of variables declared and the number of the references made to them [16]. The data access complexity metric measures the relation between the number of external data stores (files, databases, tables, etc.) access operations and the number of operations as a whole. It is based on the Card Input/Output Metric [17]. The interface complexity metric measures the relation between the number of components, modules or classes and the number of interactions between them. This is an extension of the Sallie Henry fan-in/fan-out metric [18].

The control flow complexity metric is actually the McCabe metric normalized to a rational scale. It measures the relation between edges and nodes of a control flow graph [3]. The decisional complexity metric is based on the McClure metric which measures the relation between decisions plus nesting levels and the number of statements [19]. The branching complexity metric measures the relation of the number of GOTO branches, perform subroutine calls and method invocations to the number of methods, subroutines and labels. Finally, the language complexity metric measures the relation of data operands declared and statement types used with the data operand frequencies and the statement types used with the statement type frequencies. It is based on Halstead's volume metric [4].

How complexity metrics are used depends on the goals of the measurement project. One could be predicting errors, assessing maintainability or, as was the case here, estimating effort. In estimating effort the complexity of a system is used to adjust the size of that system. Adjusted Size = Raw Size * Complexity.

3.3 Quality Metrics

The third dimension of software is quality. Quality is basically a question of the relationship between how a product is – the Ist - and how it should be – the Soll.

$$quality = \frac{Ist}{Soll}$$

The problem is in defining how software should be. Quality is the hardest measurement to take, since there are many qualities to consider – static quality, dynamic quality, internal quality and external quality. The ISO-9126 prescribes eight quality characteristics:

- Functionality
- Reliability
- Security
- Usability
- Performance
- Reusability
- Portability
- Maintainability.

Of these eight quality characteristics, only the last three are static and internal, i.e. they can be taken from the software itself without executing it. However, these three are abstract notions and as the standard suggests, they need to be broken down into lower level operational metrics which can be readily measured. Only portability can be measured directly. Reusability has to be extended by convertibility and modularity. Modularity combines a high degree of cohesion and a low degree of coupling with a minimum size of the individual modules [20]. Convertibility measures the number of directly translatable statements relative to the total number of statements [21].

Maintainability is the most abstract of all the metrics. It encompasses not only modularity but also flexibility and conformity. Flexibility implies that the code is not tied to a particular usage. It can be readily changed to be used in a different mode. In order to archive that it must be free of hard-coded data. So here it is necessary to compare the number of occurrences of hard-coded data with the number of data in all. Conformity is, on the other hand, the degree to which the code abides to the coding convention, i.e. its uniformity. Finally, in order to be reliable, code must be well tested which implies it should be testable. A component is testable when it has a minimum number of paths to test and a minimum number of control parameters to be set. Thus, the number of paths relative to the number of statements and the number of control parameters, i.e. predicates, relative to the number of all parameters determine the testability. This testability metric has been introduced by the author in a previous paper [22]. The eight quality characteristics of the code measured by the *SoftAudit* Tool are:

- portability (as the degree of independence from the environment)
- reusability (as the degree of independence of individual code units from one another)
- convertibility (as the proportion of readily convertible statements)
- modularity (as the product of cohesion and coupling)
- flexibility (in terms of proportion of hard-coded data)
- conformity (in terms of adherence to the prescribed coding rules)
- testability (in terms of paths and predicates relative to code volume)
- maintainability (as the weighted average of all qualities adjusted by the complexity)

In estimating effort quality is used to adjust the raw size. In development an above average quality increases effort. In maintenance and migration an above average quality decreases effort. Thus, Adjusted Size = Raw Size * Complexity * Quality.

4 Software Measurement Tools

The measurement of such a large mass of code is impossible without the aide of automated tools. One needs tools for every language and for every purpose. For this project not one tool but a whole family of tools was required – a Software Measurement Workbench. The software measurement workbench of the author consists of measurement tools for no less than 12 different programming languages, 8 interface description languages, 6 database description languages, 3 job control

languages plus a metric database tool and a calculation tool. Without having had all of those tools available it would have been impossible to make this measurement project. Tools are a prerequisite for all software engineering activities, but in particular for measurement and test.

4.1 SoftAudit – The Tool for Analyzing Code and Collecting Metrics

The tool *SoftAudit* measures the sizes, complexities and qualities defined in the previous section of the following:

- The procedural languages Assembler, PL/I and COBOL
- The 4th generation languages Delta, APS, CSP, EasyTrieve, Natural and ABAP,
- The object-oriented languages C++, C# and Java
- The database schemas of IMS, CODASYL, ADABAS, DB2, Oracle and SQL-Server
- The conventional map definitions of CICS, IMS-DC and Natural
- The modern interface definitions of HTML, XML, XSL, WSDL and IDL
- The job control procedures for IBM-VM, IBM-OS and UNIX Script.

There is a parser for each language as well as a unique set of counting rules and a set of coding rules to be adhered to. The development of this PC-based tool set goes back to the early 80ies when the author first developed a static analyzer for analyzing Assembler, PL/I and COBOL code on the mainframe [SnMe85]. Since then the tools have been continually evolved and enhanced. The tools for measuring database and map descriptions were added as a result of reengineering projects in Switzerland in the early 1990's. The 4th Generation Language analyzers were made in the last 80ies to deal with that emerging market and the object-oriented analyzers were developed in the late 1990ies to measure and check the object-oriented languages C++ and Java [23]. The C#, HTML, XML and WSDL analyzers have been added since 2000. Measurement of such a large mass of code is impossible without the aide of.

4.2 SoftEval – The Tool for Evaluating the Metrics

The tool *SoftEva*l is responsible for creating and populating a metric database. Its first function is to set up a relational database with 10 tables - one for each software artifact measured

- requirement document
- UML diagrams
- program source code
- interface source code
- database schemas
- test cases
- test results
- defect reports
- change requests
- project productivity reports.

There is a table for each software product measured with a line for each subsystem of that product. The columns are the individual metrics.

The second function is the editing function which allows the user to edit the metric data. The user can overwrite existing metric counts and alter existing names but he may not add new entries. They have to be imported.

The third function is to generate a series of diagrams and reports upon request

- histograms to depict the relation of subsystems to one another
- pie charts to depict the proportions of subsystems to the whole
- fish grate diagrams to depict the varying degrees of complexities and qualities for each subsystem
- distribution charts to depict deficiency and error frequency
- dashboards to depict the status of the individual systems in terms of selected key metrics
- rankings of systems by size, complexity and quality
- comparisons of systems and versions of systems depicting the degrees of change in size, complexity and quality.

4.3 SoftCalc – The Tool for Calculating Project Effort

The tool *SoftCalc* is dedicated to producing cost estimations. It imports metric data from the same operational tools as with SoftEval but only that data required to make cost estimations for different project types using various estimation methods. The eight methods supported are:

- COCOMO-I for estimating maintenance projects
- COCOMO-II for estimating maintenance, migration and reengineering projects
- Data-Point for estimating development and migration projects
- Function-Point for estimating development and migration projects
- Object-Point for estimating development, maintenance and reengineering projects
- Use-Case-Point for estimating development and maintenance projects
- Test-Case-Point for estimating maintenance, migration and test projects
- Error Projection for estimating test projects.

Like *SoftEval*, *SoftCalc* has a relational database with a set of related tables. Each of the estimations methods has its own influence and productivity tables. There are common tables for product quality goals, project risks and project resources. These tables have to be built up by the estimator.

The tables filled by the imported data are the process, i.e. use-case table, the object table, the interface table, the component table and the test case table. These tables exist for each project to be estimated.

- the use-case table is filled from the use cases extracted from the requirement documents
- the object table is filled from the objects extracted either from the requirement documents or the design documents or the database schemas

- the interface table is filled from the interface definitions extracted either from the requirement documents, the design documents or the interface description sources
- the component table is filled from the SoftAudit source program analysis
- the test case table is filled with the test case data imported from the test case analyzer.

Once the tables have been populated, *SoftCalc* can carry out any of the eight estimations listed out above and produce the time and effort, the quality rate achievable and the degree of reliability of the estimate. It is then up to the estimator to choose which, if any, of the estimates he would like to use.

5 Software Measurement Process

The process for measuring the code base in this project proceeded in eight steps executed by two persons over a three month period:

- setting up the directories
- scanning for user specific language extensions
- setting the measurement parameters
- running the source analysis
- importing the metrics into the metric database
- evaluating the metric data
- importing the metrics into the calculation metric database
- calculating and documenting the estimations

5.1 Setting Up the Directories

The measurement process began with the collection and classification of the more than 93,000 source members. Included in these sources were:

7.854	PL/I Programs
13.530	PL/I Includes
17.151	COBOL Programs
24.712	COBOL Copies
8.975	Assembler Programs
9.597	Assembler Macros
3.077	EasyTrieve Programs
2.190	EasyTrieve Copies
3.253	German Pseudo Code Texts
3.270	IMS Databases
4.416	DB2 Tables
31.476	IMS Maps
22.090	JCL Procedures
2.977	C++ Sources
1.977	Java Sources.

Since the IT organization in question included three divisions and each division had to be measured separately, there had to be a main directory for each division with a sub directory for each source type, containing all of the sources of that type. This was to be another measurement by subsystem within each division. There were altogether more then 400 subsystems. Since each subsystem could contain different languages there was a separate directory for each division in with the subdirectories were the subsystems containing sources of different types.

The reason for the different directory structures lies in the way *SoftAudit* aggregates the metrics. The metrics are aggregated at the level of the lowest directory – the module level, at the level of the next lowest directory – the component level, as well as at the system and product level. There is no aggregation at the language level, since the tool was designed to process one language at a time. If an aggregation by language is required, then the source must be processed by language. If an aggregation by component and system is required, then the sources must be ordered by component and system. For this project it meant that the sources had to be processed twice, once by language and once by component. Just setting up the directories and adding the proper extensions to the source names required two weeks time.

5.2 Scanning the Code for User Specific Language Extensions

There is no such thing as a standard language, especially not in Germany where almost every organization takes pride in developing its own generators and frameworks. This means that the host language is supplemented by user defined statements. In procedural languages like PL/I, COBOL and in particular, Assembler, these user defined statements are referred to as macros. They usually fulfill some technical function such as database access or screen manipulation, which is often used. In object-oriented languages such as CPP, C# and Java they are methods or classes in the user framework which are called to fulfill similar technical functions like accessing a data base as done by the ODBC and JDBC functions, accepting and displaying screen data or handling exception conditions.

If a measurement tool is really going to measure the size, complexity and quality of a software system, then it must recognize these constructs. For this reason the tool *SoftAudit* requires a function table to be set up prior to processing the code. For each macro and framework function the user fills in the name and assigns the macro or function to one of the existing function types for data access operations, user interactions, data exchange and internal processing operations. The counting of function points is particularly dependent on this table since function points are derived from the user inputs/outputs, database accesses, reports and data exchange between systems. The other size counts are less dependent on the function table, but it does affect the interface and access complexity as well as the portability and reusability. Therefore, such a table is vital to any serious software measurement operation.

In scanning the source prior to actually processing it, *SoftAudit* recognizes the macros and external functions and lists them out. It is then up to the user to assign them a meaning. This action can result in significant manual effort, but it has do be done if the measurement is going to be meaningful. In this project the creation of the function tables required some 11 days of effort, not including the days contributed by the customer. The preliminary scanning of the 75 million lines of program code

required a day for each of the three corporate divisions Once the function tables have been established though, one per language or source type, they can be reused for all subsequent measurements. They only have to be updated in accordance with the evolution of the macros and framework functions.

5.3 Setting the Measurement Parameters

The last task to be performed before starting the actual analysis run is to set the parameters. There are three types of parameters in *SoftAudit*

- coding rules
- code limits
- metric weights.

For each programming, database and interface language, there is a set of rules taken from the coding conventions of the more than 100 organizations for which the author has worked for in the past 40 years. Certain language constructs are considered as smells, others are down right dangerous. Still others such as hard coded literals and numeric constants detract from the flexibility and maintainability of the code. *SoftAudit* has a table per language in which all of the rules for that language are listed out. Here, the user has the opportunity of clicking out rules which he does not want to have checked.

It is similar with the limits. There are standard limits to such quantities as the number of statements per module, the number of methods per class, the number of data attributes per class or module and the number of parameters per call. These limits are intended to restrict the complexity of the program, database or interface. However, the user can override the standard limits by raising or lowering them.

Finally, the user can adjust the weights of metrics. There are 8 complexity and 8 quality metrics for programs, 5 complexity and 5 quality metrics for databases, and 6 complexity and 6 quality metrics for interfaces. Normally the metrics all have the same weight. However, the user has the opportunity here to adjust the weights. A metric can be assigned a weight of zero, normal, high or critical. If the weight is zero, it is not computed at all, if it is high then it weights twice as much as normal and if it is critical it weights four times more. In this way, the user can manipulate the complexity and quality ratings.

The setting of the parameters in this project was done in two steps. In the first step the customer company was given tables to fill out. In the second step the tables were used to set the parameters in the tool. The latter step took no more than a day, but the first step took several days, mainly because of the unfamiliarity of the customer personnel with the rules and limits.

5.4 Running the Source Analysis

The fourth step in the measurement process is the actual source processing itself. The tool operator selects individual source files or whole directories of source file. The sources can all be of one language or they can be of mixed languages. The processor takes one source file at the time, parses it, counts the quantities and checks the rules. It then adds the counts to the aggregate tables for components, subsystems and the

product as a whole as well as for that particular language. After the end of each module, component and subsystem it computes the complexities and qualities for that unit. At the end of the run, it does the same for the product as a whole and for each language.

The rule violations are written out in a deficiency report for each module. Thus at the end of a processing run, the user is left with a set of deficiency reports , one for each module, a set of metric reports, one for each module, component, subsystem, product and language. In addition, there is an XML metric export file for each subsystem and another one for the product data as a whole.

In this project three different PC workstations were used in parallel to process the sources. On the one workstation the mainframe program sources of all three corporate divisions were analyzed. On the other workstation the database schemas, map descriptions, job control procedures and the client/server sources were processed. On a third workstation the metrics were aggregated. The processing of the mainframe sources required by far the most time due to the large amount. For the one corporate division with close to 40 million lines of code the run lasted more than 8 hours. For the other two divisions with approximately 12 million lines of code apiece the runs lasted some 5 hours. The processing of the other source types could be done within a day. Thus, processing all 75 million lines of code required some two days. Unfortunately, due to missing and extra sources, the runs had to be repeated twice. For one division the run had to be repeated a third time because of a program error in the tool.

5.5 Importing Metrics into the Metric Database

If the user wants to set up a metric database as was the case in this project, he must import the XML metric files from the source analysis into the metric database. This is done with the tool *SoftEval*. *SoftEval* allows the user to allocate a database for a software product and then select XML files to import from. The data from these files is then taken to populate the metric tables. There is a table for each metric type – requirements, design, code, interface, database, test case, test, defect report, change request and project productivity. In this measurement project, only the code, the user interfaces and the databases were measured, so only these tables were filled with the metrics taken from the source code. This required less than a day.

5.6 Evaluating the Metric Data

In the sixth step the metric data was evaluated in a number of ways selected by the user. There were both product and system evaluations. For each product, all systems contained therein were ranked by size, complexity and quality. Pie charts and histograms were produced to depict the relation of the application systems to one another. In addition, a manage dashboard was displayed for each system depicting its complexity, quality, conformity and reusability. Of particular importance in this project was the ranking of the individual systems. They were ranked by size, complexity and quality. The creation of these graphics and reports took no more than two days.

5.7 Importing Metrics into the Calculation Database

Since one suspected goal of this project was to estimate the costs of potential project types, it was necessary to also import the metrics collected from the source analysis into the calculation database. The metrics for the calculations are only a subset of the metrics which go into the metric database. Thus, this set was done in parallel to the importing of the metrics to the metric database. The reason for separating them at all is to allow users to calculate project costs without establishing a complete metric database. The price for that is the consistency of the two data bases. In this project this was not a problem, since the metrics were taken from the same source and were not edited.

5.8 Calculating Possible Project Costs

The final step in this measurement project was to calculate the costs of maintaining, renovating, migrating or integrating the software products under evaluation. It was also decided to recalculate the costs of a new development, just for the sake of comparison, but for systems of this magnitude, redevelopment is not a serious option. As came out of the calculation, the costs of redevelopment would go into thousands of man years. This estimation is only useful to determine the value of a software system in terms of its costs.

However, renovation, migration and integration are real alternatives. Using the COCOMO-II, Function-Point and Data-Point methods, it was possible to estimate the costs of each of these three strategies with three different estimates. In some cases the estimates were highly scattered, but in most they were clustered, indicating that the calculation data was consistent. Rather than have a single effort estimation, a range was calculated with a lower and an upper bound of the effort required [23]. The calendar time required was derived from the effort using the COCOMO-II time equation with a variable schedule compression factor. These multiple calculations were presented to corporate management together with the size, complexity and quality evaluations.

6 Conclusion and Lessons Learned

The ultimate question for any measurement project is whether the result justifies the effort. For this project 40 person days in two calendar months were budgeted. In the end it cost 52 person days and required an additional calendar month. This resulted in a month delay for the customer and a net loss to the contractor of 12 person days. There were several reasons for this overflow. New parsers had to be developed, the measurement process control had to be reengineered to accommodate multiple languages within the same component and the measurement had to be repeated because of parsing errors. The requirements also changed as the customer learned what he should expect from such a massive measurement project. This is not the first measurement project the author has conducted. There have been many over the past 15 years starting with the Dutch Telecom in 1994 and including the Swiss Telecom, the German Telecom, the German stock exchange, the Schufa, the Bavarian Pension Fund, DebeKa Insurance, the Raiffeisen Bank and the Schwäbisch-Hall Home

Savings and Loan. The question always arises as to what motivates a user to make such a one time measurement. It may be to assess his software inventory prior to an outsourcing contract, to plan the costs of a migration or to compare one set of systems with another. The problem is that the person doing the measurement seldom knows what the customer has in mind, since this is a strategic secret. However, without knowing the goal, it is not possible to work according to the goal-question-metric method [24]. One simply measures everything possible and hopes that some of the measurements will be of use to the customer.

As witnessed by this project a lot of work is involved just to tell the user that his system has 21,843,516 statements or 152,399 function-points, that his system complexity is 0,498 and that his system quality is 0,623. These numbers will mean nothing to the customer unless he is able to see what consequences they have for his future planning. If he can see that they affect the costs of system maintenance, that they determine the costs of system migration or integration, and that they are related to the number of errors that occur in production, he begins to have a benefit. So it is up to those who commission the measurement to establish a relationship between the measurement results and the goals of their IT Management. Such goals could be inventory control, cost calculation, error prediction, problem area analysis or benchmarking. These goals need to be specified before the measurement project is started and monitored throughout the project. Of course the measurement tools have to be flexible enough to allow adapting the metrics to the goals defined. That prerequisite is only partially fulfilled by the tools used here.

In summarizing it can be said that there is much to be done in educating software managers to understand software metrics and how they relate to their goals. There is also much to be done to improve the quality and utility of measurement tools. Finally software measurement should not be a one time project every few years. It should be a permanent ongoing process within the IT department supported by tools and based upon a persistent metric database [25].

References

1. ISO/IEC: Software Product Evaluation: Quality Characteristics and Guidelines for their use, ISO/IEC Standard 9126, International Standards Organization, Genf (1994)
2. Zuse, H.: A Framework of Software Measurement. de Gruyter Verlag, New York (1998)
3. McCabe, T.: A Complexity Measure. IEEE Trans S.E. 2(6), 308 (1976)
4. Halstead, M.: Elements of Software Science, p. 79. Elsevier Pub., New York (1977)
5. Kemerer, C., Chidamber, S.: A Metrics Suite for Object-Oriented Design. IEEE Trans. S.E. 20(6), 476 (1994)
6. Albrecht, A., Gaffney, J.: Software Function, Source Lines of Code and Development Effort Prediction: A Software Science Validation. IEEE Transactions on Software Engineering 9(6), 639 (1983)
7. Welker, K., Oman, P., Atkinson, G.: Development and Application of an automated Source Code Maintainability Index. Journal of Software Maintenance 9(3), 127 (1997)
8. IEEE: Software Engineering Standards, Product Standards, Vol. 3. IEEE Computer Society Press, Los Alamitos (1999)
9. Hughes, B.: Practical Software Measurement. McGraw-Hill, Maidenhead (2000)

10. Moore, J.W.: Software Engineering Standards – A User's Road Map. IEEE Computer Society Press, Los Alamitos (1998)
11. Bush, M., Fenton, N.: Software Measurement – A conceptual Framework. EC-Esprit Project 2348, Report 2, South Bank University, London (1990)
12. Sneed, H.: MetKit Metric Data Model, EC-Esprit Project 2348, Report 9, SES GmbH, Munich (1991)
13. Sneed, H.: Applying Size, Complexity and Quality Metrics to an object-oriented Application. In: ESCOM Conference Proceedings, Hercmoncieux, GB, p. 92 (1999)
14. Dumke, R., Foltin, E., Koeppe, R., Winkler, A.: Softwarequalität durch Meßtools, p. 198. Vieweg Verlag, Braunschweig (1996)
15. Chapin, N.: A Measure of Software Complexity. In: Proc. of NCC, p. 995 (1977)
16. Elshof, J.: An Analysis of Commercial PL/I Programs. IEEE Trans. S.E. 2(3), 306 (1976)
17. Card, D., Glass, R.: Measuring Software Design Quality, p. 23. Prentice Hall, Englewood Cliffs (1990)
18. Henry, S., Kafura, D.: Software Structure Metrics based on Information Flow. IEEE Trans. on S.E. 7(5), 510 (1981)
19. McClure, C.: Managing Software Development and Maintenance, van Nostrand Reinhold, New York, p. 82 (1981)
20. Myers, G.J.: Software Reliability – Principles and Practices, p. 92. John Wiley & Sons, New York (1976)
21. Sneed, H.M.: Metriken für die Wiederverwendbarkeit von Softwaresystemen. Informatikspektrum 6, S18–S20 (1997)
22. Sneed, H., Jungmayr, S.: Produkt- und Prozessmetriken für den Softwaretest. Informatikspektrum, Band 29(1), 23 (2006)
23. Sneed, H.: Software-Projektkalkulation, p. 159. Hanser Verlag, München (2005)
24. Basili, V., Caldiera, C., Rombach, H.D.: Goal Question Metric Paradigm. Encyclopedia of Software Engineering, 528 (1994)
25. Ebert, C., Dumke, R.: Software Measurement, p. 471. Springer, Berlin (2007)

Improving Quality of Functional Requirements by Measuring Their Functional Size

Sylvie Trudel[1] and Alain Abran[2]

[1] CRIM/R&D, Montreal, Canada
sylvie.trudel@crim.ca
[2] École de Technologie Supérieure – Université du Québec/Dept. of Software Engineering and Information Technologies, Montreal, Canada
alain.abran@etsmtl.ca

Abstract. For many years, the software industry has been applying different types of reviews on their requirements documents to identify and remove defects that would otherwise propagate in the development life cycle, leading to rework and extra cost to fix at later phases. An inspection is a review technique known to be efficient at identifying defects but, like any other review technique, it does not guarantee that all defects are found. Requirements documents are also used as input for the measurement of the software size for estimation purposes; when carrying this measurement process, practitioners have often noticed defects in the requirements.

This paper reports on a research project investigating the contribution of the measurers in finding defects in requirements documents. More specifically, this paper describes an experiment where the same requirements document was inspected by a number of inspectors as well as by a number of measurers; the number and types of defects found by both inspectors and measurers are compared and discussed. For this experiment, the measurers used the COSMIC – ISO 19761 to measure the functional size and find defects. Results show significant increase in defects identification when both inspection and functional size measurement are used to find and report defects.

Keywords: Functional requirements, COSMIC, FSM, Functional size measurement, inspection, review.

1 Introduction

Software requirements are written to describe software that will be later developed. Requirements fall usually into two categories: functional requirements and non functional requirements. The functional requirements describe system functionalities while the non functional ones, also called technical requirements and quality requirements, describe required system attributes such as performance, security, and reliability. The focus of the research reported here is on functional requirements.

Requirements impact all phases of the software life-cycle as shown in Figure 1. Therefore, ambiguous, incomplete and incorrect requirements may negatively impact

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 287–301, 2008.

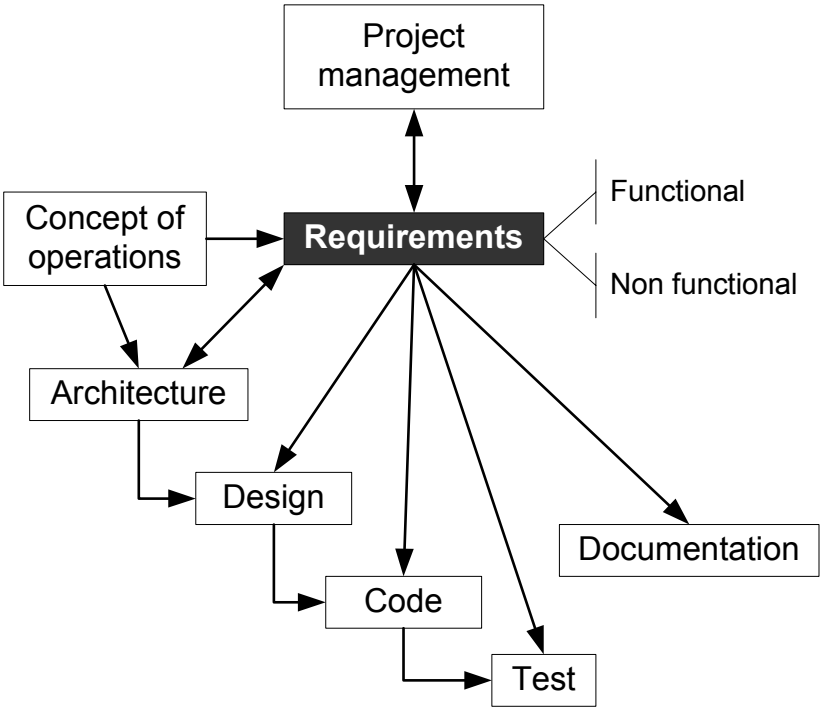


Fig. 1. Requirements usage in software development life-cycle phases

all phases if not detected early enough to be corrected; when not found, those will typically require rework to rectify work done in previous phases of the life cycle.

To minimize rework effort and cost for fixing defects at later phases in the development life-cycle, many organizations apply various review techniques on their requirements documents. Review techniques typically include a set of rules to help requirements authors and reviewers in achieving quality attributes of their requirements, such as those stated in the IEEE-Std-830-1998 [1]: "Correct", "Unambiguous", "Complete", "Consistent", and "Verifiable".

An inspection [2] is a review technique known to be efficient at identifying defects but, like any other review technique, it does not guarantee that all defects are found. To increase the efficiency and effectiveness for finding defects in software artefacts, it is recommended that organizations use several verification techniques.

Review efficiency represents the ability of a software team to identify and remove defects in an artefact. Review efficiency can be measured in number of defects found in that artefact at review time compared to the total number of defects found in the whole software project for which the origin can be traced back to that same artefact. Review effectiveness corresponds to the average effort spent in identifying critical defects.

In the early phases of the development life cycle, these same requirements documents are also used as an input for the measurement of the software functional

size, typically for estimation purposes. When carrying this measurement process for estimation purposes, measurers often observe a number of defects in the functional requirements.

This contribution of measurers at finding defects in requirements documents has not been investigated yet and has not been yet documented in the literature as a review technique, even though it is a current measurers practice.

The use of software measurement as a review technique raises a number of questions, such as:

1. Is functional size measurement (FSM) more efficient than inspections for identifying defects in functional requirements?
2. Is functional size measurement (FSM) more effective than inspections for identifying defects in functional requirements?
3. Would it be of value-added to inspections, either for efficiency or effectiveness, if a measurer's role is included?

This paper reports on an experiment carried out to investigate the third question. The experiment reported here was conducted in November 2007 with both industry and academic experts participating to the MENSURA-International Workshop on Software Measurement held in Palma de Majorque (Spain).

For the experiment reported here, the same requirements document was inspected by three inspectors as well as by four measurers. For this experiment, the measurers used the COSMIC – ISO 19761 to measure the functional size and find defects.

1.1 The Inspection Method

The inspection method used in the experiment is an adaptation from Gilb and Graham's work [3][1] . This inspection method contains seven steps as shown in Figure 2.

1.2 The COSMIC Method

Functional size measurement (FSM) is a means for measuring the size of a software application, regardless of the technology used to implement it.

The COSMIC functional size measurement method [4] is supported by the Common Software Measurement International Consortium (COSMIC) and is a recognized international standard (ISO 19761 [5]). In the measurement of software functional size using COSMIC, the software functional processes and their triggering events must be identified.

The unit of measurement in this method is the data movement, which is a base functional component that moves one or more data attributes belonging to a single data group. Data movements can be of four types: Entry (E), Exit (X), Read (R) or Write (W). The functional process is an elementary component of a set of user requirements triggered by one or more triggering events, either directly or indirectly, via an actor. The triggering event is an event occurring outside the boundary of the

[1] This inspection method has been applied successfully in a Canadian organization more than 2000 times over a four years period and numerous times in other Canadian organizations over the last seven years.

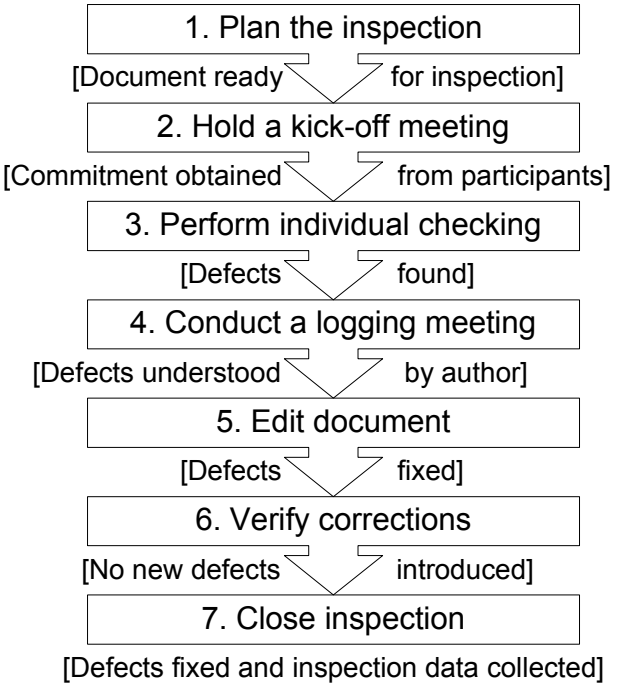


Fig. 2. Steps of the inspection method

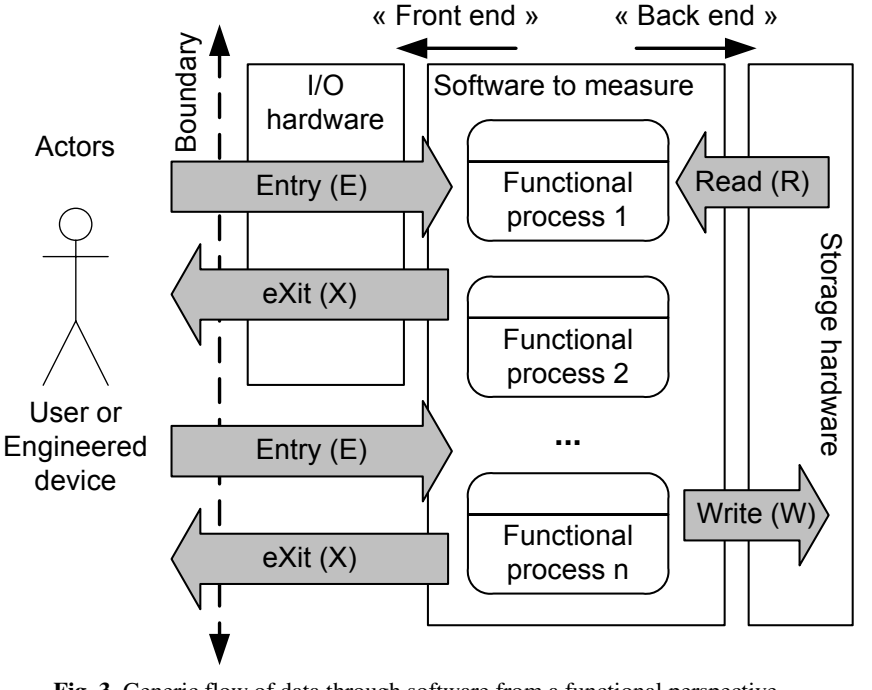


Fig. 3. Generic flow of data through software from a functional perspective

measured software and initiates one or more functional processes. The sub processes of each functional process constitute sequences of events, and a functional process comprises at least two data movement types: an Entry plus at least either an Exit or a Write. An Entry moves a data group, which is a set of data attributes, from a user across the boundary into the functional process, while an Exit moves a data group from a functional process across the boundary to the user requiring it. A Write moves a data group lying inside the functional process to persistent storage, and a Read moves a data group from persistent storage to the functional process. See Figure 3 for an illustration of the generic flow of data groups through software from a functional perspective.

2 The Experiment

2.1 Purpose and Objective of the Experiment

The main objective of the experiment was to assess the efficiency and effectiveness of the COSMIC method as a method for finding defects in software functional requirements.

The purpose was to perform an experiment involving industry experts, some of whom would be skilled in measuring functional size with the COSMIC method and others who would either be skilled in inspecting requirements or be knowledgeable on what is a well written software functional requirement. Special care was taken to get experienced practitioners in FSM and experienced inspectors and requirements writers in participating to this experiment.

2.2 The Requirements Document

The software requirements specification (SRS) document that was chosen for the experiment was compliant with IEEE-Std-830 for its structure and content. This SRS was also compliant with UML 2.0 [6] for the use case diagram, the behavioural state machine, and use case details.

1) SRS overview

The SRS was entitled "uObserve Software Specification" [7] and had 16 pages of descriptive text in English and approximately 2900 words.

Section 1 of the SRS describes the introduction, purpose and scope, project objectives, background information, and references. Section 2 provides a high-level description of the system to develop, the list of features and functions (included and excluded), user characteristics, and assumptions, constraints, and dependencies. Section 3 list all specific requirements, beginning with the user interface and its prototype, the hardware interfaces, followed by functional requirements (section 3.2), and quality requirements (section 3.3).

2.3 The Participants

1) The inspectors

Three inspectors participated in the experiment. They all cumulate years of industry practice as software practitioners where they had to write and verify software requirements. The first inspector had 8 years of industry practice, she then worked 3 years in a research facility, and she has been teaching software engineering for 4 years

during which she participated in industry research projects. The second inspector had over 6 years of industry practice, and has been teaching software engineering for more than 13 years. The third inspector has over 8 years of industry experience and was registered in Ph.D. program in software engineering.

2) The measurers
Four measurers participated in the experiment. They were all COSMIC Certified Entry Level practitioners [8] and were experienced in functional size measurement. All of them were active members of the COSMIC Measurement Practice Committee.

2.4 The Experiment Steps

The experiment consisted in the following steps applied prior to and during the experiment.

1) Prepare experiment
   a) Prepare material

Prior to the workshop experiment, the chosen SRS was reviewed by a peer to remove most spelling and syntax defects that were injected by the translation of the original requirements document from French to English. Other minor issues were also identified and fixed.

The inspection training material (e.g. templates and procedures) used in this experiment comes from the industry practice of one of the researcher [9].

The experiment material included the chosen SRS, a presentation of the inspection method, the detailed seven steps method, the inspection form for data collection, a defined set of rules, a defined set of inspector roles, definitions for defect and issue types [10] (see Table 1), and definitions for defect categories (see Table 2).

Improvement suggestions and questions are considered as issues, not as defects. However, a question may later be transformed into a critical or minor defect, depending upon the nature of the question and its related answer.

Table 1. Definitions for defect and issue types

Type	Definition
Critical or major	Defect that is likely to cause rework, or prevent understanding or desired functionality.
Minor	Information is wrong or incomplete but does not prevent understanding.
Spelling/Syntax	Spelling or syntax error.
Improvement	The product can stay as is but would be better if the improvement suggestion is implemented.
Question	Any question to the writer of the product.

Table 2. Definitions for defect categories

Category	Definition
Functional	Defect related to functional requirements or functional description of the system.
Non functional	Defect not related to functional requirements or to functional description of the system.
Undetermined	Defect that cannot be categorized into Functional or Non functional when first identified.

Table 3. Required inspector roles and their definition

Role	Definition
Logic	Focus on logical aspects of the product under inspection, making sure that "everything holds together" (catchall role).
User	Focus on the user or customer point of view (checklist or view point role).
Tester	Focus on test considerations (testability, test requirements, order of testing and order of development for parallel testing, and so on).
Standards	Verify conformity to agreed standards (quality assurance role).

Defect categories were defined for analysis purposes, since measurement should primarily be dealing with the functional description of the system to develop.

b) Call for participation

The Call for participation to the experiment was included within the Call for participation to the MENSURA-IWSM-2007, knowing that there was a mix of industry and academic experts. All participants who volunteered for the experiment had previously participated in peer reviews.

2) Provide training on the inspection method

A two-hour training session was provided to all participants on the inspection method, the rules, the roles, and the behaviours to expect and to avoid from inspection participants (inspection leader, author, and inspectors).

3) Perform inspection

a) Plan the inspection

For this experiment, the inspection leader was not given any inspector role: the inspection leader's role was to make sure the process would be followed.

The required roles were chosen from the list of roles (see Table 3). Assigning several inspector roles aims to maximizing defect identification since many perspectives are being applied.

The inspection scope was defined as sections 2 and 3 of the SRS, which size was measured at 2600 words. Thus, planned individual checking effort was set to 1 hour and 45 minutes (105 minutes) based on an inspection rate of 5 pages per hour (one page=300 words). The source documents were the SRS (section 1 – Introduction) itself and applicable standards (IEEE-Std-830 and UML 2.0).

Two inspection modes were defined in the inspection method: "parallel" or "serial". In "parallel" mode, every inspector has his own copy of the artifact to inspect and they perform their individual checking at the same time. In "serial" mode, only one copy of the artifact to inspect is carried from the first inspector to the last on the inspectors list, allowing inspectors to learn from identified defects by previous inspectors. Because of time constraints of the workshop experiment, the "parallel" inspection mode was applied.

The inspection planning was done prior to the workshop session and required 15 minutes of effort.

b) Hold a kick-off meeting

A brief overview of the SRS was provided to the inspectors. Instructions were given to inspectors to categorize every identified defect into F, N, or U, along with the defect type (see TABLE I).

The Logic role was assigned to inspector #1. The User role was assigned to inspector #2. The Tester and Standards roles were both assigned to inspector #3. All inspectors agreed to play their assigned roles.

From that moment, measurers were asked to leave the room to provide a quiet environment to inspectors.

The inspection kick-off duration was 10 minutes with a total of five participants: three inspectors, one inspection leader, and the writer of the SRS.

c) Perform individual checking

Inspectors performed their individual checking, playing their assigned roles the best they could. Defects and issues were identified and noted on the copy of the SRS of each inspector, along with their respective types and categories. Inspectors stopped the checking activity when they were convinced they had completed the required verification.

Next, each inspector compiled the number of defects per type and reported this data on the inspection form. They also measured their checking effort and compiled it on the inspection form.

d) Perform functional size measurement

The inspection training provided guidance on defect types and categories to measurers, whom attended the session as well. When the writer of the SRS handed a printed copy of the SRS to each measurer, measurers were asked to apply the COSMIC measurement method and to identify any defect and issue, along with its respective type and category.

While inspectors were checking, measurers began the FSM activity, identifying, categorizing, and providing a type for any defect and issue, which may have slowed down measurement.

Each measurer identified functional processes, data groups, and related data movements. Data movements were added to provide the functional size of every functional

process. Functional size of each functional process was added to provide the functional size of the system. Once measurers completed the FSM activity, the following data was reported on their inspection forms: effort to measure and identified defects, number of defects per type, and software functional size.

e) Conduct a logging meeting

When both inspectors and measurers had completed their activities, a logging meeting was conducted with the inspection leader, and the inspectors to describe every identified defect and issue. The objective of the logging meeting was for the writer of the SRS to understand all these defects and issues to be able, at the edit phase, to apply an appropriate correction and, if required, a type reclassification (e.g. from Question to Minor or Critical).

The logging meeting duration was one hour (60 minutes), during which all inspectors explained identified defects, focusing on Critical and Minor defect types. The Spelling/syntax type was voluntarily skipped since explanation did not seem relevant. Measurers described only some of their identified defects and the effort it required was negligeable.

At the end of the logging meeting, all SRS hand-written copies were given to the author and experimenter. Later, these copies were scanned individually into a PDF file for verification purposes.

4) Compile experiment data

a) Defects and issues log

Defects and issues were logged on a spreadsheet with the following parameters:

- Location (page #, section #, paragraph #, and line #);
- Description;
- Type (C, M, S, I, or Q);
- Category (F, N, or U);
- Number of inspectors (if more than one identified the same defect or issue);
- Inspectors initials;
- Number of measurers (if more than one identified the same defect or issue);
- Measurers initials;
- Status (Open, Fixed, or Closed); and
- Comment from the researcher.

When appropriate, the researcher reclassified the defect type and category. When two participants identified the same defect with a different type, the defect type that had the most impact was logged (i.e. Critical over Minor).

The spreadsheet allowed filtering data to ease analysis.

b) FSM detailed data

The following FSM detailed data was captured in a spreadsheet:

- Functional process;
- Data groups;
- For each measurer:
    i. Data movements per data group;
    ii. Size per data group;

iii. Size per functional process;
iv. System functional size.

c) Effort data

Effort spent per participant for the checking activity and the measuring activity was entered in a spreadsheet. The effort unit of measure was one minute. Effort spent for the other steps of the inspection method was entered separately.

5) Review experiment data with participants

Individual data were isolated and sent to each participant for review and approval. Inspectors reviewed their defects and issues log, and the number of defects and issues per type against the scanned copy of their hand-written commented SRS. Measurers reviewed the same data as inspectors plus their detailed FSM data. Data were hidden from one another to avoid any bias or influence. This step was made to ensure that data analysis would be performed with unbiassed data.

At the time this paper was written, 5 participants out of 7 had sent review feedback with either minor changes or no comment.

6) Analyze experiment data

In industry, FSM is more likely to be performed by a single measurer. Therefore, experimenting with four measurers represents four different experiments.

From the inspection point of view, the industry applies from three to five inspectors for a single inspection of a requirements document. Therefore, data from all three inspectors was combined in a single set of experiment data.

3 The Results

3.1 Inspection Results

a) Identified defects

The log per participant contained a total of 227 defects and issues, as shown in Table 4.

Table 4. Number of defects and issues by type per participant, including duplicates

		Defects			**Issues**		**Total**
Type		**C**	**M**	**S**	**Q**	**I**	
Inspectors	Insp #1	20	24	10	1	5	**60**
	Insp #2	10	28	2	0	6	**46**
	Insp #3	7	5	0	0	2	**14**
Measurers	Meas #1	5	1	8	2	1	**17**
	Meas #2	4	2	5	0	0	**11**
	Meas #3	8	14	6	1	0	**29**
	Meas #4	15	11	20	2	2	**50**
Total:		**69**	**85**	**51**	**6**	**16**	**227**

Table 5. Number of unique defects and issues by type, by category

		Defects			Issues		Total
Type		**C**	**M**	**S**	**Q**	**I**	
Category	F	37	55	17	5	4	**118**
	N	21	20	19	1	12	**73**
Total:		**58**	**75**	**36**	**6**	**16**	**191**

Table 6. Number of unique defects and issues by inspectors

		Defects			Issues		Total
Type		**C**	**M**	**S**	**Q**	**I**	
Category	F	19	39	6	1	3	**68**
	N	17	15	6	0	10	**48**
Total:		**36**	**54**	**12**	**1**	**13**	**116**

Several defects and issues were identified by more than one participant. A total of 191 uniquely identified defects and issues were recorded, as shown in TABLE V, by both inspectors and measurers.

Table 6 shows the 116 uniquely identified defects and issues found by inspectors. Measurers also identified 16 of these 116 defects and issues.

b) Effort spent and effectiveness

Inspectors spent an average of 57 minutes for the checking activity (minimum=55 minutes, maximum=60 minutes). The planned effort per inspector was 105 minutes. Total effort spent by the three inspectors was 170 minutes.

Effort for identifying defects requires not only the checking effort but also effort from previous steps and the logging meeting step [11]. Table 7 provides a summary of effort spent by the inspection team to identify defects.

The effectiveness of an inspection can be calculated as the total effort to identify defects divided by the number of critical defects. In this inspection, the effectiveness is 535 minutes / 36 unique critical defects = 15 minutes per critical defect.

Table 7. Effort spent by inspection team

Inspection step	**Duration**	**# Participants**	**Effort**
Plan the inspection	15 min	1	15 min
Hold a kick-off meeting	10 min	5	50 min
Perform individual checking	--	3	170 min
Conduct a logging meeting	60 min	5	300 min
Total:			**535 min**

3.2 Measurement Results

a) Functional size
Functional size measures in COSMIC Function Point (*cfp*) showed some variations among measurers (see Table 8). Some of these variations in the sizes obtained might be due to defects in the SRS; the sources of these variations will be analyzed in a later phase of this research project.

Table 8. Functional size per measurer in *cfp*

	Functional size	Average	Standard deviation
Meas #1	62	**59**	**3.3**
Meas #2	55		
Meas #3	61		
Meas #4	57		

b) Identified defects
Measurers have identified between 9 and 39 functional and non functional defects and issues that inspectors did not identify, as shown in Table 9, including duplicates (i.e. defects found by more that one measurer).

Table 9. Number of defects and issues found by measurers only

		Defects			**Issues**		Total
Type		**C**	**M**	**S**	**Q**	**I**	
Measurers	Meas #1	3	1	5	2	1	**12**
	Meas #2	3	2	4	0	0	**9**
	Meas #3	6	13	4	1	0	**24**
	Meas #4	10	8	17	2	2	**39**

Nevertheless, it was expected that measurers would find a majority of functional defects since the FSM activity focuses on functional description of the software. Table 10 presents the defects found by the measurers when considering only functional defects, including duplicates.

Given these figures, what would have been the value-added of individual measurers over the inspection team?

Table 11 provides the number of critical and minor defects, as well as critical only defects, identified by measurers and their relative value-added over the functional defects found by the inspection team.

Table 10. Number of functional defects found by measurers only

		Defects			Issues		Total
Type		C	M	S	Q	I	
Measurers	Meas #1	3	1	4	1	1	**10**
	Meas #2	3	2	3	0	0	**8**
	Meas #3	6	13	3	1	0	**23**
	Meas #4	6	3	6	2	0	**17**

Table 11. Value added of measurers over inspection team

	Critical & Minor	Value-added	Critical only	Value-added
Inspection team	58	--	19	--
Meas #1	4	7%	3	16%
Meas #2	5	9%	3	16%
Meas #3	19	33%	6	32%
Meas #4	9	16%	6	32%

All four measurers individually added value to the inspection team efficiency. The increase of defects identification was ranging from 7% to 33% when critical and minor defects are considered. The value-added was even higher when considering only critical defects, ranging from 16% to 32%.

c) Effort spent

Measurers have spent an average of 57 minutes for the measurement activity, including defect identification, as shown in Table 12.

Table 12. Effort spent by measurers in minutes

	FSM effort	Average	Standard deviation
Meas #1	49	**57**	**13.4**
Meas #2	45		
Meas #3	60		
Meas #4	75		

On average, a measurer took the same amount of effort for performing FSM and identifying defects and issues than an inspector for performing the individual checking step.

In this experiment, the effectiveness of the FSM activity for finding defects cannot be isolated since the effort was spent focusing on sizing the software application.

No time limit was imposed on measurers. However, during the measurement activity, measurers had move to an open space of the conference facility and complained that the noise level had slowed down their measurement pace.

4 Discussion and Future Work

FSM results typically provides the functional size of the software, allowing a development team or project manager to use this input for estimation and benchmarking purposes. Another important value-added data comes out from this measurement activity is the identification of defects not found by a team of inspectors.

The experiment results demonstrated a value-added on inspection efficiency when having a measurer who raises issues while measuring the functional size. Adding measurement over inspection allowed identifying from 16% to 32% of new critical functional defects, in less effort than the planned individual checking effort. Of course, inspectors do not provide functional size data as it is not part of an inspection method.

Inspectors spent 54% of the planned effort for their individual checking. If the planned checking effort would have been spent totally, inspectors might have found a larger number of defects and issues.

Measurers participating in this experiment may have been over experienced and other less experienced measurers may lead to different results. This will require further experimentation to verify this.

Further work includes other experiments with industry requirements documents that may or may not be compliant with IEEE-Std-830 and UML 2.0.

Acknowledgments

The authors thank participants to the experiment.

The inspectors: Maya Danava, Ph.D., software engineering professor at University of Twente, Netherlands; Mohamad Kassab, test specialist, Oz Communication, www.oz.com, and Ph.D. graduate student, Concordia University, Canada; and Olga Ormandjieva, Ph.D., software engineering professor at Concordia University, Canada.

The measurers: Harold Van Heeringen, measurement expert, Sogeti, www.sogeti.nl, Netherlands; Luca Santillo, measurement expert, Agile Metrics, www.agilemetrics.it, Italy; Charles Symons, software measurement expert, founder and joint project leader COSMIC, England; and Frank Vogelezang, measurement expert, Sogeti, Netherlands.

References

1. IEEE Computer Society, IEEE-Std-830-1998, IEEE Recommended Practice for Software Requirements SpeciÞcations, New York, NY (June 1998)
2. Wiegers, K.: Peer Reviews in Doftware: A Practical Guide. Addison-Wesley, Boston (2001)
3. Gilb, T., Graham, D.: Software Inspections, pp. 13–20. Addison-Wesley Professional, Reading (1993)

4. Abran, A., et al.: COSMIC-FFP Measurement manual: the COSMIC implementation guide for ISO/IEC 19761:2003, version 2.2, Common Software Measurement International Consortium (January 2003)
5. International Organization for Standardization, ISO/IEC 19761:2003, Software engineering – COSMIC-FFP – A functional size measurement method (February 2003)
6. Arlow, J., Neustadt, I.: UML 2 and the Unified Process, 2nd edn. Addison-Wesley, Reading (2005)
7. Trudel, S., Lavoie, J.M.: uObserve Software Specification. École de Technologie Supérieure, Montreal (2007)
8. GÉLOG, COSMIC Entry Level Practitioners Certificate Holders, `http://www.gelog.etsmtl.ca/cosmic-ffp/entry_level_holders.html`
9. Trudel, S.: Software Inspections Workshop. CRIM, Montreal, Canada (2007)
10. Canadian Department of National Defence, Defect type definitions (unpublished)
11. Stewart, R., Priven, L.: Revitalizing Software Inspections. In: Montreal Software Process Improvement Network (SPIN), Canada (February 6, 2008)

Implementing Software Project Control Centers: An Architectural View

Jens Heidrich and Jürgen Münch

Fraunhofer Institute for Experimental Software Engineering,
Fraunhofer-Platz 1, 67663 Kaiserslautern, Germany
{jens.heidrich,juergen.muench}@iese.fraunhofer.de

Abstract. Setting up effective and efficient mechanisms for controlling software and system development projects is still challenging in industrial practice. On the one hand, necessary prerequisites such as established development processes, understanding of cause-effect relationships on relevant indicators, and sufficient sustainability of measurement programs are often missing. On the other hand, there are more fundamental methodological deficits related to the controlling process itself and to appropriate tool support. Additional activities that would guarantee the usefulness, completeness, and precision of the resulting controlling data are widely missing. This article presents a conceptual architecture for so-called Software Project Control Centers (SPCC) that addresses these challenges. The architecture includes mechanisms for getting sufficiently precise and complete data and supporting the information needs of different stakeholders. In addition, an implementation of this architecture, the so-called Specula Project Support Environment, is sketched, and results from evaluating this implementation in industrial settings are presented.

Keywords: Software Project Control Center, Measurement, QIP, GQM.

1 Introduction

Many companies still have problems in setting up effective and efficient mechanisms for project control. According to a study by the Standish Group [1], even though the general expertise in project management and techniques has improved over the last years, around 50% of the projects are still over budget and schedule. Unfortunately, this figure has not changed since the first CHAOS report results were published in 1994. In order to overcome deficits in controlling a software development project, companies have started to introduce so-called software cockpits, also known as Software Project Control Centers (SPCC) [2] or Project Management Offices (PMO) [3], for systematic quality assurance and management support. Software cockpits centrally integrate all relevant information for monitoring and controlling purposes. For instance, a project manager can use them to get an overview of the state of a project, control schedule, effort, and cost, and a quality assurance manager can use them to check the quality of the software produced. An important success factor is that control centers can be customized to the specific goals, organizational characteristics and

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 302–315, 2008.

needs, as well as the concrete project environment. Implementing such control centers is a challenging task. It is not (only) a question of having a customizable generic tool, but primarily a question of finding suitable indicators for controlling the project and having concrete guidelines on how to introduce project controlling functionality and general measurement capabilities into an organization. That is, comprehensive methodological support is needed for successfully setting up and using mechanisms for quantitative project control. There are several approaches for deriving indicators and metrics from high-level measurement goals. One of the most popular ones is the Goal Question Metric (GQM) paradigm [4], which supports explicit definition of measurement goals and has a structured approach for deriving corresponding metrics via a set of questions that help to determine whether the measurement goal has been achieved. However, with respect to project control, a comprehensive methodology that supports the whole life cycle including planning and setting up project control mechanisms, using them continuously for controlling a development project, systematically analyzing the deficits of the used mechanisms, and packaging experiences in order to continuously improve project control, is usually missing.

Specula is a state-of-the-art approach for project control. It interprets and visualizes collected measurement data in a goal-oriented way in order to effectively detect plan deviations. The control functionality provided by Specula depends on the underlying goals with respect to project control. If these goals are explicitly defined, the corresponding functionality is composed out of packaged, freely configurable control components. Specula was mainly developed in the context of the public German research project Soft-Pit (No. 01ISE07A) and makes use of the Quality Improvement Paradigm (QIP) [5] for integrating project control activities into a continuous improvement cycle. Furthermore, the GQM approach is used for explicitly specifying measurement goals for project control. The basic methodology and an extensive usage example are described in [6]. The approach was evaluated as part of industrial case studies in the Soft-Pit project, where the prototypical implementation was used to provide project control functionality for real development projects. Results of the first two iterations can be found in [7] and [8]. A summary of success factors extracted so far from applying the approach and our experience in setting up and using quantitative project control are presented in [9].

The aim of this paper is to talk about how to concretely implement a control center addressing all relevant goals with respect to project control following the general Specula methodology. Section 2 gives an overview of typical problems that have to be addressed when implementing control centers and summarizes strengths and weaknesses of existing methods and technical approaches. Section 3 illustrates a conceptual architecture for control centers and the basic functionality that has to be provided. Moreover, the basics concepts of the Specula approach are summarized, including the conceptual model and the basic methodology for setting up and using the project control functionality. Section 4 presents the Specula Project Support Environment tool, which was implemented based on this architecture and was used as a kind of product line for flexibly composing the needed project control functionality for the different case studies conducted. Section 5 summarizes the results from evaluating the approach, including some lessons learned with respect to the concrete tool prototype used. Section 6 concludes with a brief summary and outlook on future work.

2 Project Control in Research and Practice

Setting up a set of suitable mechanisms for project control and applying them correctly during the lifetime of the project is a challenging task. Especially for small and medium-sized enterprises, it is difficult to establish mechanisms for quantitative project control due to the limited resources for setting up appropriate processes and analyzing data. If expert knowledge for setting up a customized measurement program for project control is missing or its implementation seems to be too costly, project control is often done using out-of-the-box functionality as provided by standard project control tools instead of defining and controlling specific measurement goals. Typical dashboards provide only a fixed set of indicators and visualizations with quite simple customization mechanisms; a higher-level quality model that helps to analyze and interpret the indicators in the context of a clearly defined measurement goal is usually missing. There exists a huge set of specific tools for controlling different aspects of cost, time, and quality, but no single point of project control that covers all relevant aspects for controlling the project is provided. In research, several approaches exist that provide partial solutions to the problem of effective and efficient control of development processes. Deficits can be seen especially with respect to supporting purpose- and role-oriented project control by flexibly combining control mechanisms. An overview of these approaches can be found in [2]. The indicators that are used for project control should be derived in a systematic way from the project goals [10] (using, e.g., GQM). Some indicator examples can be found in [11].

In practice, approaches from the business intelligence area, such as Pentaho (*www.pentaho.com/*) or Jaspersoft (*http://www.jaspersoft.com/*) can be used to construct software dashboards. They are able to connect to different data sources, extract the relevant information, and store this information in a database. They offer different analysis engines for providing dashboard visualizations and report generation. They provide an open interface for extending their capabilities towards integrating project control functionality. However, methodological support for systematically deriving the right control mechanisms for a project and organization based on context information and organizational goals is usually missing. Pentaho and Jaspersoft could be customized to address different aspects of project control. Most commercial dashboards in the area of software project control focus on a certain aspect, like technical quality, schedule adherence, or performance indicators. A more holistic approach addressing all aspects relevant for project control is not in the focus of these kinds of dashboards. For instance, the CAST AD Governance Dashboard (*http://www.castsoftware.com/*) focuses on code quality and provides a customizable set of indicators for analyzing and assessing different quality aspects with respect to technical quality. However, it is not clear how to select appropriate indicators that fit the specific goals of a project or a certain organization.

3 Conceptual Architecture of Control Centers

Specula (Latin for watchtower) is an approach for constructing control centers in a goal-oriented way. It was developed focusing on extensibility (with respect to the control functionality provided), customizability (with respect to the context in which the control functionality is applied), and reusability (with respect to the functionality

offered). It composes the project control functionality out of packaged, freely configurable control components. Specula consists of the following components:

- a *conceptual model* formally describing the interfaces of reusable control components for data collection, data interpretation, and data visualization,
- a *methodology* of how to select control components according to explicitly stated goals and customize the SPCC functionality,
- a *conceptual architecture* for implementing software cockpits, and
- a *prototype implementation* of the conceptual model, including a construction kit of predefined control components.

The conceptual model as well as the basic methodology and a high-level conceptual architecture were presented in [6]. In this section, we will summarize some basics with respect to the model (Section 3.1) and the methodology (Section 3.2) needed to understand the basic structure of the architecture and the corresponding prototype implementation. After that, all elements of the conceptual architecture (Section 3.3) will be discussed. As the focus of this paper is on implementing control centers, the prototype implementation will be discussed in a separate section (Section 4).

3.1 Conceptual Model

The central component of the Specula conceptual model is a *visualization catena (VC)*, which defines components for automatically and manually collecting measurement data, processing and interpreting these data, and finally visualizing the processed and interpreted data. The whole visualization catena has to be adapted in accordance with the context characteristics and organizational environment of the software development project currently being controlled. Fig. 1 gives an overview of all VC components and their corresponding types. Specula distinguishes between the following five components on the type level from which a concrete VC is instantiated:

- *Data types* describe the structure of incoming data and data that is further processed by the VC. For instance, a time series (a sequence of time stamp and corresponding value pairs) or a project plan (a hierarchical set of activities having a start and end date and an effort baseline) could be logical data types.
- *Data access object packages* describe the different ways concrete data types may be accessed. A special package may be used, for instance, to automatically connect to an effort tracking system or bug tracking database.
- *Web forms* describe a concrete way of managing measurement data manually, involving user interaction. A web form refers to certain data types that are needed as input. For instance, in order to enter effort data manually, one needs the concrete activities of the project for which the effort is tracked.
- *Functions* represent a packaged control technique or method, which is used to process incoming data (like Earned Value Analysis, Milestone Trend Analysis, or Tolerance Range Checking). A function needs different data types as input and produces data of certain data types as output.
- *Views* represent a certain way of presenting data, like drawing a two-dimensional diagram or just a table with a certain number of rows and columns. A view visualizes different data types and may refer to other views in order to create a hierarchy of views.

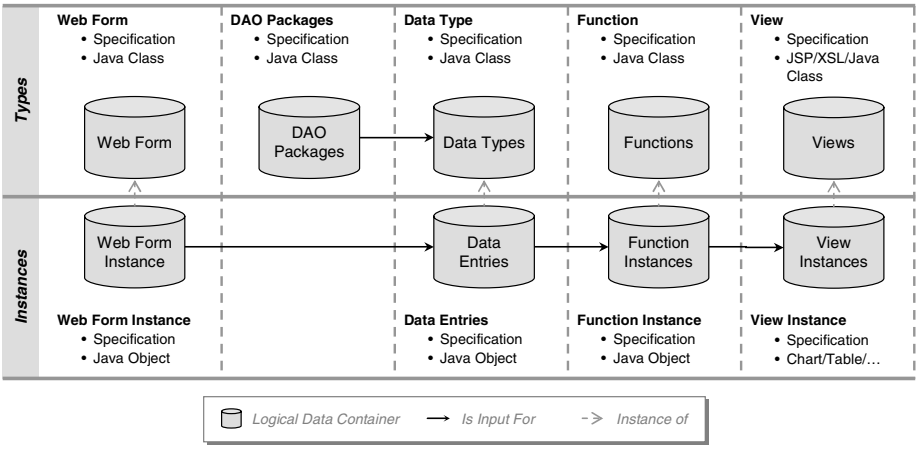


Fig. 1. Basic structure of the Specula repository. Each type has a formal specification (e.g., inputs, outputs, parameters) and a corresponding implementation. A type may be instantiated. Such instances also have a formal specification (e.g., the concrete data that is used as input or that is produced, or the concrete parameter setting that is used) and use the implementation of the corresponding type to perform their tasks (e.g., reading, aggregating, or visualizing data).

A VC is instantiated from the types described above by using the following components on the instances level:

- *Data entries* instantiate data types and represent the concrete content of measurement data that are processed by a control center. External data must be read-in or imported from an external location, or manually entered into the system. Each external data object has to be specified explicitly by a data entry containing, for instance, the start and end times and the interval at which the data should be collected. In addition, the data access object package that should be used to access the external data has to be specified.
- *Web form instances* provide web-based forms for manually managing measurement data for data entries.
- *Function instances* apply the instantiated function to a certain set of data entries. A function instance processes data and produces output data, which could be further processed by other function instances or visualized by view instances.
- *View instances* apply the instantiated view to a certain set of data entries. A view instance may refer to other view instances in order to build up a hierarchy.

A visualization catena consists of a set of data entries, each having exactly one active data access object for accessing incoming data, a set of web form instances for managing the defined data entries, a set of function instances for processing data, and finally, a set of view instances for visualizing the processing results.

Fig. 2 presents excerpts of the visualization catena for a practical course held at the University of Kaiserslautern. The catena contains all control components needed for ensuring that the actual effort of the project stays below the planned effort for all activities. The upper part of the figure shows the instances and the lower part the

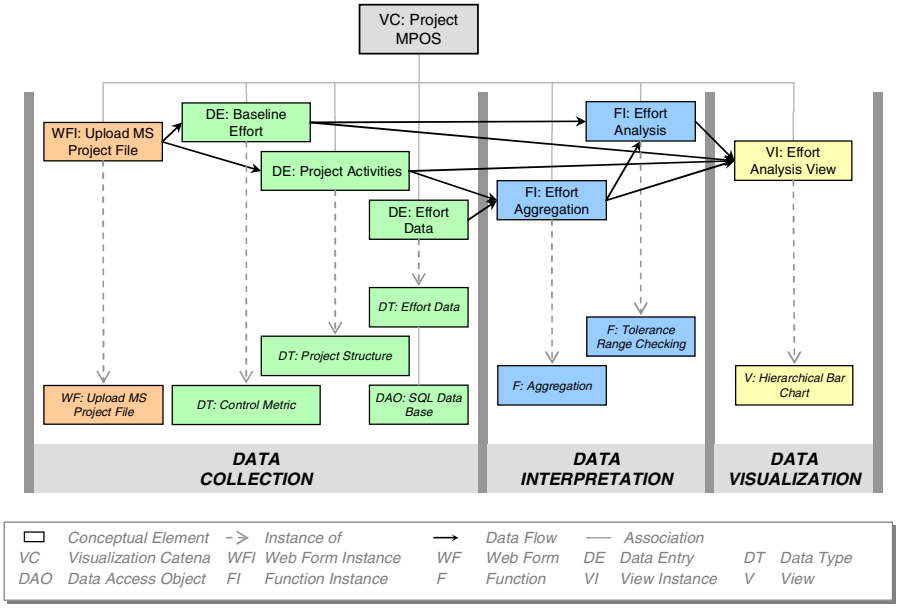


Fig. 2. Example of a visualization catena. A visualization catena is composed of web form instances, data entries, function instances, and view instances. These elements are instances of corresponding types: web forms, data types, functions, and views.

instantiated types, that is, the reused control components. The data collection area specifies three data entries: one representing the baseline effort per activity (instance of a control metric data type), one representing the hierarchy of project activities (instance of a general project structure data type), and one representing the actual effort data per project team member and project activity (instance of a general effort table and accessed via an SQL data connector). For collecting the project activities and the planned effort, a web form instance is defined, which imports the information from an MS Project file. The data processing area defines two function instances: one for aggregating effort data across the defined activities (in order to compute the actual effort per project activity) and one for comparing the actual effort with the planned effort per activity (making use of a tolerance range checking function). The data visualization area defines one view instance that visualizes the actual effort, the planned effort, and the computed effort deviation along all project activities using a bar chart that is able to drill down into the effort data along the hierarchy of project activities.

3.2 Methodology

Specula makes use of the QIP for introducing an improvement-oriented software project control cycle. QIP is used to implement a project control feedback cycle and make use of experiences gathered for reusing and customizing control components. GQM is used to drive the selection process of finding the right control components according to explicitly defined measurement goals. The different phases that have to

be considered for setting up and applying project control mechanisms can be characterized as follows (see [6] for a more extensive discussion and examples):

- *I. Characterize Control Environment:* First, project stakeholders characterize the environment in which project control shall be applied in order to set up a corresponding measurement program that is able to satisfy all needs.
- *II. Set Control Goals:* Then, measurement goals for project control are defined and metrics are derived determining what kind of data to collect. In general, any goal derivation process can be used for defining control objectives. For practical reasons, we focus on the GQM paradigm for defining concrete measurement goals.
- *III. Goal-oriented Composition:* Next, all control mechanisms for the project are composed based on the defined goals; that is, control techniques and visualization mechanisms are selected from a corresponding repository and instantiated in the context of the project that has to be controlled. This process is driven by a measurement plan that clearly defines which indicators contribute to specific control objectives, how to assess and aggregate indicator values, and how to visualize control objectives and intermediate results.
- *IV. Execute Project Control Mechanisms:* Once all control mechanisms are specified, a set of role-oriented views is generated for controlling the project. When measurement data are collected, the control mechanisms interpret and visualize them accordingly, so that plan deviations and project risks are detected and a decision-maker can react accordingly. If a deviation is detected, its root cause must be determined and the control mechanisms have to be adapted accordingly. This, does, for example, require data analyses on different levels of abstraction in order to be able to trace causes of plan deviations.
- *V. Analyze Results:* After project completion, the resulting visualization catena has to be analyzed with respect to plan deviations and project risks detected in time, too late, or not detected at all. The causes for plan deviations and risks that were detected too late or that were not detected at all have to be determined.
- *VI. Package Results:* The analysis results of the control mechanisms that were applied may be used as a basis for defining and improving control mechanisms for future projects (e.g., selecting the right control techniques and data visualizations, choosing the right parameters for controlling the project).

3.3 Conceptual Architecture

[9] presents a more abstract representation of the conceptual SPCC architecture. The view presented here is more sophisticated and addresses visualization catena handling in much more detail ([2] presents an earlier version of this view). The SPCC architecture is organized along three different layers. The *information layer* gathers all information and data that are essential for the functionality, for instance measurement data from the current project, experiences from previous projects, and internal information, such as all available Specula instances and types. The *functional layer* performs all data processing activities; that is, it executes chosen function instances and composes view instances. Finally, the *application layer* is responsible for all user interactions;

that is, it provides the resulting information of the functional layer to an SPCC user and receives all incoming user requests. Each layer consists of several conceptual elements that provide the essential project control functionality. An overview of the architecture is presented in Fig. 3. In the following, an overview of the essential conceptual elements covered by the conceptual architecture is given.

- *Repository Management Unit:* The repository management unit provides access to a repository containing reusable parts of the underlying conceptual model: VC types (data types, DAO packages, functions, views, and web forms) and VC instances (data entries, function instances, view instances, and web form instances).
- *EB Management Unit:* The experience base management unit provides access to an experience base (EB). One EB section provides project-specific information, such as the measurement data of the current project, the project goals and characteristics, and the project plan. The other EB section provides organization-wide information, such as quality models (e.g., as a basis for data prediction) and qualitative experience (e.g., to guide a project manager by providing countermeasures). The EB management unit organizes access to an experience base by providing mechanisms for accessing distributed data sources (in case of distributed development of software artifacts), for validating incoming data, and for integrating new experiences into the (organization-wide) EB. The EB management unit accesses (external) data sources and creates logical data containers (data entries) that may be used by the data processing and packaging units.
- *Customization Unit:* The customization unit is in charge of creating the visualization catena that is responsible for controlling a software development project. That is, it needs to instantiate the corresponding types from the SPCC repository. The types have to be selected based on the goals and characteristics of the project. Specula uses a GQM plan for specifying measurement goals, questions, and metrics. Based on the information provided there, suitable types are selected from the repository and instantiated. If the SPCC repository does not provide appropriate components, new types have to be defined and stored in the repository that may be reused by future projects. The VC instances have to be customized according to the project specifics. This includes setting the required input and all parameters needed for using the specific type.
- *Data Processing Unit:* The data processing unit receives the visualization catena from the customization unit. It analyzes all function instances, that is, it determines input and output information, the function's implementation, and the relationships to other function instances. If a function instance is based on other function instances, an appropriate execution sequence is computed. During execution of the chosen function instances, the data processing unit receives data entries from the EB management unit, respectively already processed data from a previously executed function instance. The results of a function instance have to be updated if an underlying data unit or function instance result has changed. The results of all executed functions are delivered to the presentation unit for data visualization.
- *Presentation Unit:* The presentation unit receives the visualization catena from the customization unit. It analyzes all view instances, that is, it determines the relationships between the view instances and the function instance outputs and data entries that have to be used to create the corresponding visualization. If a view instance is

based on other view instances, an appropriate creation sequence is computed. A view instance has to be updated if the underlying data has changed. The results of all views are delivered to the user communication unit.

- *Packaging Unit:* The packaging unit is responsible for all information that is fed back into the system by the user. This includes all external data provided via web form instances. It summarizes all experiences gained from the usage of an SPCC, adapts them according to the needs of future projects (i.e., generalizes the information units), and delivers them to the EB management unit for integration into the respective section of an experience base.
- *User Communication Unit:* The user communication unit determines the access granted to a specific user. That is, it permits a certain user to access the results of a certain set of function instances or a certain set of view instances. Furthermore, it provides a graphical user interface (GUI) for administering the SPCC (e.g., user management) and for the goal-oriented selection of VC components (via the customization unit). This includes selecting appropriate data types, functions, views, and web forms, and adapting the resulting visualization catena. Last, it provides access to the generated visualizations (delivered by the presentation unit) and manages interaction with them (e.g., drilling down or filtering data).

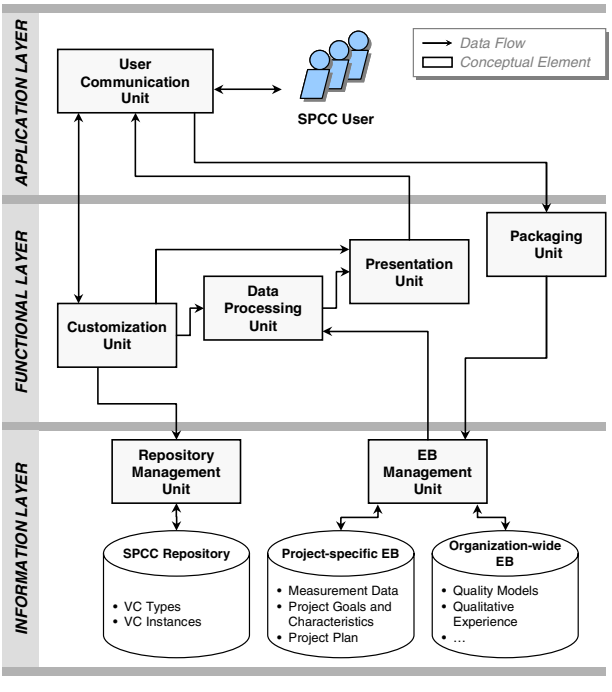


Fig. 3. General conceptual architecture of the Specula project control center. The different conceptual elements represent logical tasks that need to be performed by a control center in order to access information, interpret and analyze it, and communicate with an SPCC user.

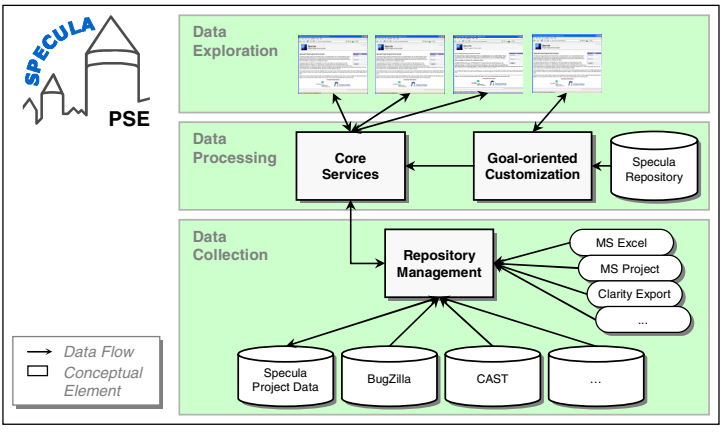


Fig. 4. Technical high-level architecture of the Specula PSE tool. The tool provides capabilities for collecting data, processing them according to the specified visualization catena, and finally visualizing and exploring them.

4 The Specula Project Support Environment Tool

Large parts of the conceptual architecture presented above are implemented by the Specula Project Support Environment (PSE) tool, which completely automates the conceptual units except for parts of the customization and packaging units. The Specula PSE tool can be used as a framework for systematically composing project control mechanisms based on reusable control components; it provides a core functionality for project control and clearly defines interfaces for specifying additional modules that can be freely enhanced with respect to specific needs. Customization includes specification of types, instances, and administration information (users and groups), implementation of data access object packages for accessing different repositories, implementation of data types for defining logical data containers, implementation of functions for processing measurement data, implementation of views for displaying data, and implementation of web forms for managing (importing, exporting, adding, removing) data. Specula PSE is a web-based software implemented as a Java-Servlet and runs on top of a Tomcat web server. The tool has a classical three-layered design (as presented in Fig. 4) in correspondence to the layers of the conceptual architecture:

- The *data collection layer* deals with accessing different data sources. Project data and measurement data need to be collected automatically by accessing different existing databases, or semi-automatically by using web forms for importing data from files or for entering data manually. For instance, a data type and corresponding data access object may be specified for accessing defect data stored in a BugZilla database (*http://www.bugzilla.org/*), or a web form may be specified for importing project plan information stored in an MS Project file.
- The *data processing layer* uses the data collection layer for accessing data from different sources in a unique way, processing them according to the VC defined, and finally providing services upon the processing results. Different services are

offered for user management, checking the consistency of a VC specification, accessing data repositories and VC specifications, etc. In order to adapt the Specula PSE functionality to project goals and characteristics, a corresponding customization unit manages all control components; that is, it supports the definition of new control components, the reuse of existing components, and the parameterization of control components according to the project context.

- The *data exploration layer* uses the services of the data processing layer for providing a graphical user interface, including displaying charts and tables, managing data, administering control components, and importing/exporting data.

The process of deriving a VC from a GQM plan (including project goals and characteristics) is currently not automated by the tool and must be performed manually. In the future, this process could partly be automated depending on the degree of formality of the corresponding GQM models, interpretation models, and further contextual information. However, currently, performing this process requires a deeper understanding of the measurement program and the control components of the Specula repository that may potentially be reused for implementing the measurement program. The Specula prototype tool automates the specification and packaging of all control components of the conceptual model and is able to automatically execute the derived visualization catena. SPCC users may use the tool for collecting measurement data and for utilizing the generated visualizations for project control. Support for setting up and accessing an organizational experience base is currently also limited and restricted to managing control components. The control components contained in the Specula repository depend on the organization (and the very project that should be controlled). Some components may be more general and applicable for several companies and projects, whereas others may be very specific and implement organization-specific control strategies. This is also related to the different *kinds* of components in the repository. For instance, one control component may implement a (fairly) complex control technique (like Earned Value Analysis) and another component may just provide some simple data processing functionality for supporting other functions (like scaling a time series or converting between different data types). Specula PSE comes with a set of standard data collection forms, control techniques, and views that were used as part of case studies and may serve as a basis for adding further elements to the framework.

Fig. 5 shows the internal structure of the tool. Let us assume that the VC as shown in Fig. 2 was specified and executed by the tool. Let us also assume that the project plan was updated and a user wants to import an MS Project file by using a web form instance of the VC. Web form instances are implemented as Java Server Pages (JSP). Based on the VC specification, the tool automatically creates a web page for uploading the file. During uploading, the content of the file is analyzed and then transformed into so-called transport objects (implemented as Java classes). The three layers of the system are connected via these transport objects. They actually contain the data that is collected, processed, and visualized. The content of the transport objects is stored in the system using the repository service. According to the VC, the file contains (a) a list of project activities and (b) the baseline effort for all activities. The VC manager recognizes that data belonging to two data entries of the VC was updated and automatically initiates an update of the corresponding function instances (and other control components affected). In our case, the implementation of the function instance,

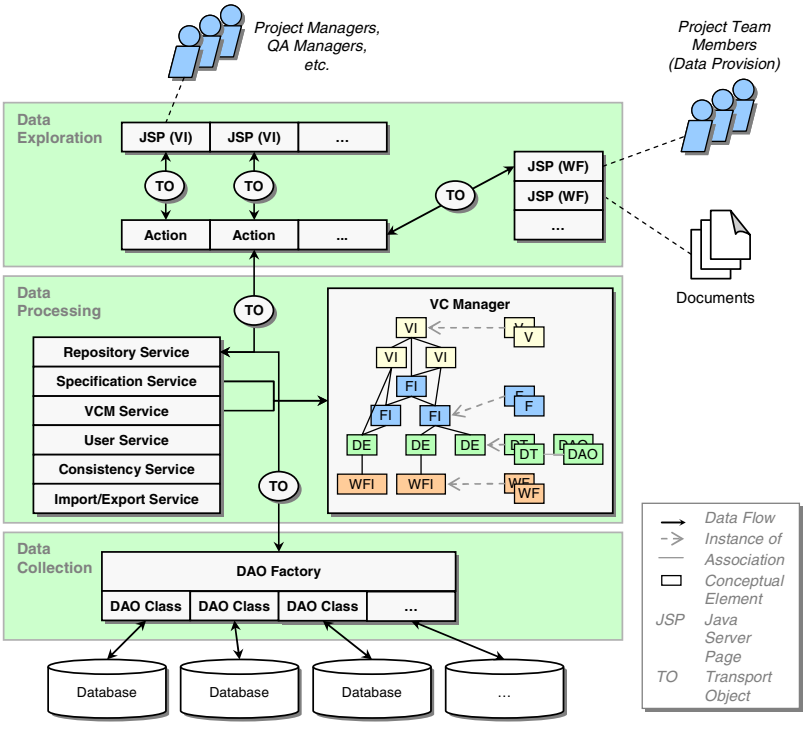


Fig. 5. Internal structures and data flow of the Specula PSE tool. The data collection layer accesses different data repositories and sends the information via so-called transport objects to the data processing layer and the VC manager, which in turn processes the data according to the VC specification and passes it on to the data exploration layer for visualization. Transport objects may also flow back to the back end layer if data is imported from other sources.

comparing the actual effort against the baseline effort, is invoked and the effort analysis is performed according to its specification. That is, a corresponding Java class is instantiated, provided with the necessary data for performing the effort analysis, and invoked accordingly.

View instances can be implemented in three different ways. The most common way is to provide a JSP page containing some graphical illustrations (using, e.g., JFreeChart, *http://www.jfree.org/*) and tables. Based on the VC specification, the tool automatically creates a web page for displaying the graphical representation of the view. If the project manager wants to see the effort controlling view of the example VC, the corresponding JSP page is provided with the necessary data for creating a bar chart containing the planned and actual effort per project activity. Buttons for navigating through the hierarchy of project activities and filtering data are also provided. The required data is automatically retrieved from a project database, stored in a transport object, and delivered to the exploration layer using the repository service.

5 Evaluation Results

The approach was evaluated as part of industrial case studies in the Soft-Pit project in which the prototypical implementation was used. Results of the first two iterations can be found in [7] and [8]. In this section, we will highlight some of the evaluation results over the three project iterations. In general, people perceived the usefulness and ease of use of the Specula control center as positive (which was evaluated using the Technology Acceptance Model [12]). The results for ease of use were not as promising as the results for usefulness. This is not a surprising result given the fact that a prototype was used during the case studies. All users received a basic training in using the control center, but depending on their familiarity with such tools, the results varied. The general usefulness and ease of use also varied across the different case study providers depending on the state of the practice before introducing the Soft-Pit control center solution. We continuously improved the method for setting up the control center and provided concrete guidelines for the case study providers on how to perform concrete tasks. As a consequence, the usefulness increased continuously over the three iterations. For evaluating the efficiency of the control center, we analyzed the detected plan deviations and project risks. Overall, 18 deviations and risks were detected by the control center in the second and 21 in the third iteration. The approach was able to detect between 40% and 80% of the listed plan deviations and project risks earlier than the traditional approaches to project control used by the case study providers before introducing the Soft-Pit solution. More than 20% of plan deviations and project risks were found that would not have been detected at all without using the control center. The contexts in which the control center was applied differed quite a lot depending on the case study provider. It included small and medium-size companies as well as a large organization. The ratio of control center costs to the overall development costs varied between 11% and 14% for a team size of 7 team members and between 9% and 10% for a team size of 17 team members. This relatively high ratio might have been related to the fact that some tasks had to be performed manually, and that the evaluation period was too short, so that activities that usually have to be performed just once had a bigger impact on the overall figures.

6 Conclusions

This article presented a conceptual architecture for control centers and the Specula PSE controlling tool implementing this architecture. The approach implements a dynamic approach for project control; that is, measures and indicators are not predetermined and fixed for all projects. They are dynamically derived from measurement goals at the beginning of a development project. A context-specific construction kit is provided, so that elements with a matching interface may be combined. The qualitative benefits of the approach include: allowing for more transparent decision-making, reducing the overhead for data collection, increasing data quality, and, finally, achieving projects that are easier to plan and to control. Future work will concentrate on setting up a holistic control center that integrates more aspects of engineering-style

software development. The starting point for setting up such a control center are usually high-level business goals, from which measurement programs and controlling instruments can be derived systematically. Thus, it would be possible to transparently monitor, assess, and optimize the effects of business strategies.

References

1. Standish Group. CHAOS Summary 2008. Study, Standish Group International (2008)
2. Münch, J., Heidrich, J.: Software Project Control Centers: Concepts and Approaches. Journal of Systems and Software 70(1), 3–19 (2004)
3. Project Management Institute: A Guide to the Project Management Body of Knowledge *(PMBOK® Guide)* 2000 Edition. Project Management Institute, Four Campus Boulevard, Newtown Square, PA 19073-3299 USA (2000)
4. Basili, V.R., Caldiera, G., Rombach, D.: Goal Question Metric Approach. In: Encyclopedia of Software Engineering, pp. 528–532. John Wiley & Sons, Inc., Chichester (1994)
5. Basili, V.R., Caldiera, G., Rombach, D.: The Experience Factory. Encyclopaedia of Software Engineering 1, 469–476 (1994)
6. Heidrich, J., Münch, J.: Goal-oriented setup and usage of custom-tailored software cockpits. In: Jedlitschka, A., Salo, O. (eds.) PROFES 2008. LNCS, vol. 5089, pp. 4–18. Springer, Heidelberg (2008)
7. Ciolkowski, M., Heidrich, J., Münch, J., Simon, F., Radicke, M.: Evaluating Software Project Control Centers in Industrial Environments. In: International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), Madrid, pp. 314–323 (2007)
8. Ciolkowski, M., Heidrich, J., Simon, F., Radicke, M.: Empirical Results from Using Custom-Made Software Project Control Centers in Industrial Environments. In: International Symposium on Empirical Software Engineering and Measurement (ESEM 2008), Kaiserslautern (to be published, 2008)
9. Ciolkowski, M., Heidrich, J., Münch, J.: Practical guidelines for introducing software cockpits in industry. In: Proceedings of the 5th Software Measurement European Forum (Smef 2008), Milan, May 28-29-30, 2008, pp. 49–64 (2008)
10. Kitchenham, B.A.: Software Metrics. Blackwell, Oxford (1995)
11. Agresti, W., Card, D., Church, V.: Manager's Handbook for Software Development. SEL 84-101, NASA Goddard Space Flight Center. Greenbelt, Maryland (November 1990)
12. Davis, F.D.: Perceived usefulness, perceived ease of use, and user acceptance of information technology. MIS Quarterly 13(3), 319–340 (1990)

Towards a Comprehensive Approach for Assessing Open Source Projects

Marcus Ciolkowski and Martín Soto

Fraunhofer Institute for Experimental Software Engineering
Kaiserslautern, Germany
{ciolkows,soto}@iese.fraunhofer.de

Abstract. Open Source Software (OSS) has an increasing importance for the software industry. Similar to traditional (closed) software acquisition, OSS acquisition requires an assessment of whether quality is sufficient for the intended purpose. This includes assessing a software component's intrinsic quality, as well as its supplier's maturity (i.e., ability to deliver high quality) and sustainability (i.e., whether the supplier will continue to exist). For traditional software acquisition, established procedures are available for evaluating these aspects. These procedures need to be adapted for OSS projects, because they have no traditional supplier, but an underlying OSS community. The openness of OSS development presents both challenges and opportunities for project evaluation. In particular, a variety of data sources are available that potentially allow for in-depth analysis, but it is not clear how to use them effectively.

In this paper, we present an approach toward a comprehensive measurement framework for OSS projects, developed in the EU project QualOSS. This approach takes into account product quality as well as process maturity and sustainability of the underlying OSS community.

Keywords: Open Source quality, process assessment, process maturity.

1 Introduction

Once considered the product of a fringe movement, the strong, and ever increasing, influence of Open Source Software (OSS) on industry can hardly be denied anymore. The potentially large benefits offered by OSS, such as valuable functionality at relatively low cost and independence from a particular supplier, are now routinely taken into account by decision makers in charge of industrial software acquisition. Consequently, it is not surprising that, in recent years, OSS has made inroads into many industry branches. This trend is not likely to change in the near future.

Contrary to common belief, however, OSS adoption does not come for free. The implementation of OSS systems in an organization, as well as their use as components in larger systems, is accompanied by all manner of risks and uncertainties. Questions ranging from the appropriateness for the task at hand to the licensing issues involved must be properly addressed before a particular piece of software is selected.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 316–330, 2008.

Answering such questions is not only a difficult task in itself, but the consequences of a wrong answer may be serious.

The fact that OSS is not free of cost or risk is often used as a general argument against it. Yet, it should be taken into account that traditional, commercial-off-the-shelf (COTS) software is similar in many ways. Software acquisition, both for OSS or COTS components, is concerned with questions such as the following:

- Does the component provide the required functionality?
- Is it "good enough" for the purposes at hand?
- Will we be able to find support for it in five years from now?
- Are the licensing conditions compatible with our intended business model?

In other words, the functionality and quality of a component as well as the sustainability of its supplier(s), and the legal conditions under which it is offered are crucial aspects in both cases.

In order to answer questions such as those listed above, a number of aspects of the product and its supplier need to be evaluated. Some of these aspects are related only to the product itself, and can be measured directly on it for OSS and COTS software alike. In this respect, OSS often offers an advantage, because the source code is always available and can be readily analyzed, whereas COTS software is seldom as transparent. In fact, in recent years, OSS has often been the target of quantitative code quality analysis for both research and industrial purposes.

There are a number of relevant aspects, however, that require information beyond what product measurement can provide. For instance, development processes and organizational structures of the supplier organization must often be taken into account when evaluating software. In this respect, OSS is different from COTS because it is not provided by a supplier organization in the classical sense, but by a world-wide community of (often voluntary) developers. It may seem that such a community can hardly be evaluated with respect to its capability to deliver high software quality. Indeed, a common perception is that OSS communities normally work in an unstructured and chaotic way that cannot consistently lead to high-quality results. However, quite a few OSS projects (the Linux kernel and the Apache web server are only two among many available examples) have been consistently delivering appropriate quality over a number of years, a fact that indicates that the previous perception is generally not true.

This raises the question of how we can find out to what degree an OSS community has the potential to deliver good product quality over time. Fortunately, due to their open nature, OSS projects provide us with plenty of potential information sources, such as code, mailing lists, bug tracking systems, and versioning systems, among others. We believe that, by systematically analyzing these information repositories, it is possible to investigate many of those quality aspects that are not restricted to the product itself.

In the EU project QualOSS ("Quality of Open Source Software"), we aim at developing a comprehensive model for assessing robustness and evolvability of OSS projects. In terms of the above-mentioned acquisition questions, this addresses OSS quality ("is it good enough") and sustainability ("will be able to find support in future?"). In this paper, we present the initial prototype version of the QualOSS model.

The remainder of this paper is structured as follows: Section 2 presents some related work on process and Open Source assessment. In Section 3, we describe a comprehensive measurement framework for OSS projects that takes into account product quality as well as process maturity and sustainability of the underlying OSS community. Section 4 summarizes the results of an initial evaluation of the QualOSS model. Finally, Section 5 summarizes the paper and suggests some possibilities for future work.

2 Related Work

2.1 Process Assessments

Since the introduction of the CMM in the early 1980s, maturity-oriented process assessment models have become a central tool to determine the extent to which an organization can deliver software on time and with an acceptable quality level. Some prominent examples of such process assessment models are CMMI-DEV (Capability Maturity Model® Integration for Development [1]) and SPICE (Software Process Improvement and Capability dEtermination [2]).

The growing popularity of OSS represents a new challenge with respect to software quality assessment, since, at first sight, maturity-oriented models are not applicable to OSS. On the one hand, they seem to expect an organizational structure that is not present in most OSS communities. On the other hand, it is a widespread belief that OSS communities operate in an essentially chaotic way, and that, for this reason, no systematic development processes can be taking place during OSS development. Consequently, most casual observers would regard traditional maturity models as completely inappropriate for OSS software.

We disagree with the previous idea. The main assumption underlying process assessment approaches is that more mature processes consistently lead to higher quality products, whereas for an organization with immature processes the capacity to deliver high-quality products is unreliable and cannot be predicted. There is no reason to believe that this assumption is not valid for OSS. Concretely, we expect that a higher level of process maturity will lead to better products and more sustainable communities.

2.2 OSS Quality Assessment

In recent years, Open Source software has often been the target of quantitative of code quality analysis, mostly due to the fact that large code repositories are available. In consequence, many publications exist on (semi-) automatic analysis of code, mailing lists, bug tracking, and versioning systems.

As a reaction to the insight that not only code aspects need to be considered, assessment models for OSS projects have emerged to support potential OSS users. The most prominent examples are QSOS [7] (Qualification and Selection of Open Source software), OpenBRR [8] (Open Business Readiness Rating), and two different models called “Open Source Maturity Model” [9, 10] (OSMM). OpenBRR was built based on these two OSMM models; therefore, we will not consider them further in this paper. Both models, QSOS and OpenBRR, define a hierarchy of characteristics

and define procedures to rate the leaf characteristics. The aggregated rating is computed through weighted mean of all leaf scores.

QSOS

QSOS [7] splits its evaluation template into two kinds of sections: A generic section (including criteria that apply to all software products) and a specific section, which includes a list of the expected functionality and therefore varies according to software product family, such as groupware, CMS, database, etc. In this paper, we concentrate on the generic section, as our focus is not on evaluating functionality but on generic quality properties.

Fig. 1 shows an excerpt of the tree-level hierarchy of evaluation criteria in QSOS's generic section. Table 1 highlights the scoring procedure for two criteria.

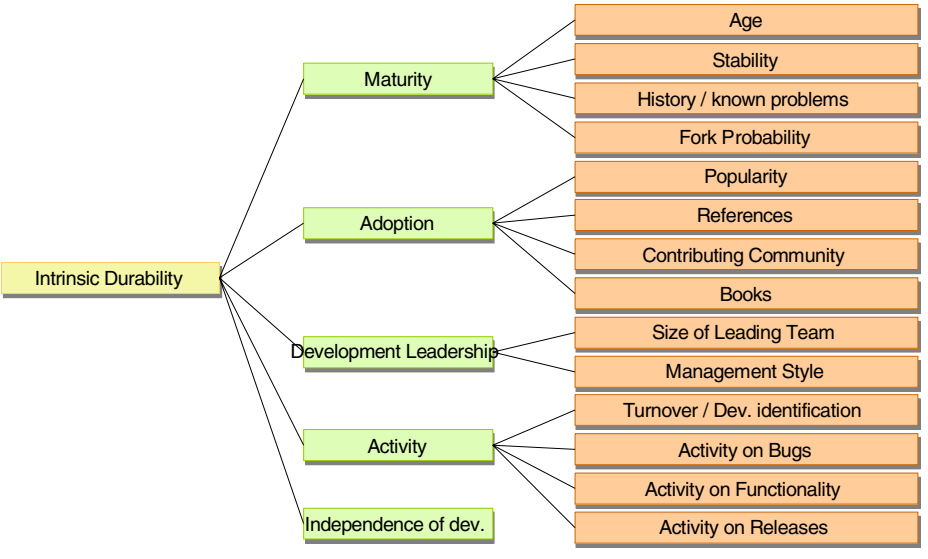


Fig. 1. Excerpt of QSOS hierarchy version 1.6. This figure shows one of the four QSOS main characteristics; the others are: "Industrialized solution", "technical adaptability", "strategy", and "providing services".

Table 1. Examples of the QSOS scoring procedure. Each of the leaf characteristic is rated on an ordinal scale from 1 to 3.

Criteria	**Score=1**	**Score=2**	**Score=3**
Age	Less than 3 month old	Between 3 months old and 3 years old	More than 3 year old
Training	No offer of training identified	Offer exists but is restricted geographically and to one language or is provided by a single contractor	Rich offer provided by several contractors, in several languages and split into modules of gradual levels

OpenBRR

The elements of the OpenBRR hierarchy are defined on metric level; for example, under the Quality category, we find "number of minor releases in the past 12 months". However, the metrics used by OpenBRR can be abstracted into quality characteristics (see Fig. 2).

Fig. 2 shows an excerpt of the tree hierarchy of evaluation criteria in OpenBRR. Table 2 highlights the scoring procedure for two criteria.

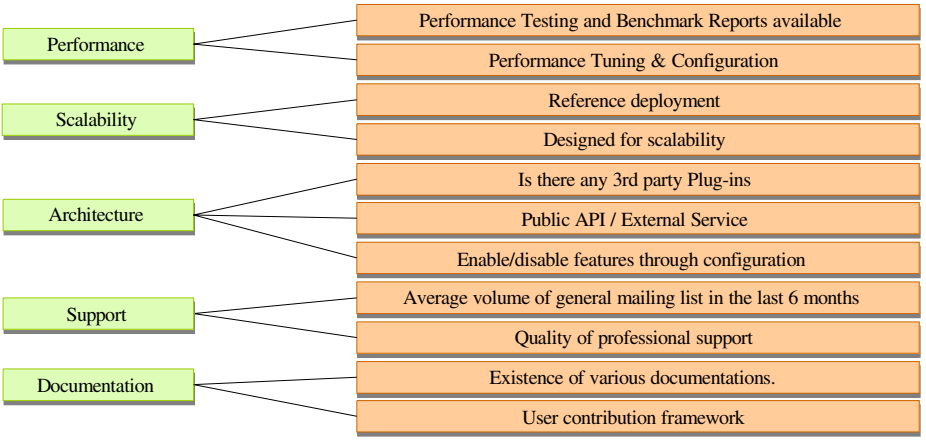


Fig. 2. OpenBRR Model Hierarchy for five of eight OpenBRR main characteristics. Others are "adoption", "community", and "professionalism".

Table 2. Examples of the OpenBRR scoring procedure. Each leaf characteristic is mapped onto an ordinal scale from 1 to 5; not all values are defined for all leaf criteria.

Criteria	**Score=1**	**2**	**3**	**4**	**Score=5**
Time for vanilla installation	>4 hours	1-4 hours	30 min – 1 hour	10-30 min	<10 min
User contribution framework	Users cannot contribute		Users are allowed to contribute		Users are allowed to contribute and contribution are edited / filtered by experts

Discussion

The QSOS and OpenBRR models take the OSS product into account (i.e., code, documentation), as well as the community. However, they only have a rudimentary process perspective, if at all. For example, QSOS considers two process criteria: QA processes (with levels *none*, *informal*, *supported by tools*), and bug/feature request tools (*none, standard tools, active use of tools*). In other words, existing OSS assessment models do not consider process maturity.

In addition, both models define their characteristics and metrics without specifying the purpose or goal of the underlying measurement, and they do not separate between

community and product aspects. In consequence, it is hard to argue why the models use the characteristics and metrics they use, and to what degree they are complete.

Deprez and Alexandre [11] compared QSOS and OpenBRR and came to the conclusion that both models – in addition to the above-mentioned issues – do not require evaluators to capture the location of the raw data used to obtain the evaluation scores; consequently, the repeatability of an assessment is unclear. Table 3 shows an overview of their findings.

Table 3. Comparison of QSOS and OpenBRR (adapted from [11])

	Strengths	Weaknesses
QSOS	• Open repository of evaluation scores for various FlOSS projects (this pushes evaluators to collaborate on evaluation and to facilitate cross validation) • Extensive list of criteria • Interesting innovating nomenclature for the tree hierarchy • QSOS methodology is versioned and evaluation mention the QSOS version used	• Ambiguous scoring rules for more than half of the criteria • Scoring procedure with 3-level scale may make decision making harder • Universality of scoring rules is not possible for many criteria • No clear reasoning for characteristics and metrics
OpenBRR	• Allows for tailoring hence better fit one's evaluation context • Clearer scoring procedure with fewer ambiguities • 5-level scoring scale for about half of the criteria • Ask evaluator to perform a quick assessment step to reduce the evaluation effort	• No open repository of evaluation (due to possible tailoring) • Does not exploit the 5-level scales for more than half of the criteria • Terminology is broad and imprecise for the top nodes in the hierarchy • OpenBRR does not seem to be versioned. However, this may be left to the evaluator • No clear reasoning for characteristics and metrics

3 The QualOSS Model Framework

The QualOSS model was originally designed to support the quality evaluation of OSS projects, with a focus on evolvability and robustness. One central, underlying assumption while defining the model has been that the quality of a software product is not only related to the product itself (code, documentation, etc.), but to the way the product is developed and distributed. For this reason, and since the development of OSS products is the responsibility of an open community, the QualOSS model takes both product- and community-related issues into account on an equal basis, and as comprehensively as possible.

The QualOSS model is composed of three types of interrelated elements: quality characteristics, metrics, and indicators. Quality characteristics correspond to the concrete attributes of a product or community that we consider relevant for evaluation (see below for an explanation of how these characteristics were chosen.) Metrics correspond to concrete aspects we can measure on a product or on its associated

community assets, that we expect to be correlated with our targeted quality characteristics. Finally, indicators define how to aggregate and evaluate the measurement values resulting from applying metrics to a product or community in order to obtain a consolidated value that can be readily used by decision makers when performing an evaluation.

The quality characteristics in the model are organized in a hierarchy of two levels that we call characteristics and subcharacteristics for simplicity. The subcharacteristics are considered to contribute in some way or another to the main characteristic they belong to. In order to define our hierarchy of quality characteristics, we relied mainly on three sources: (1) Related work on OSS quality models as outlined above, (2) general standards for software quality, such as ISO 9126 [15], and (3) expert opinion; that is, we conducted interviews among industry stakeholders to initially derive relevant criteria for the QualOSS model.

Given our emphasis on covering not only OSS products but the communities behind them, we have grouped the quality characteristics into two groups: those that relate to the product, and those that relate to the community. The rest of this section provides more details about the composition of these groups.

3.1 Product

Being open and community-based, OSS development differs, often strongly, from its commercial counterpart. Still, the products resulting from this mode of development are not, per se, essentially different from similar commercial products. For this reason, our approach is to use a rather standard set of criteria to evaluate OSS products (e.g., from ISO 9126 [15]). Also for this reason, we will not go into much detail about product evaluation in the present article. Interested readers are referred to the relevant QualOSS project deliverable [14]. Table 4 summarizes the QualOSS product quality characteristics.

As mentioned in the introduction, one advantage we have when dealing with OSS is the availability of source code, which makes it possible to apply many standard code measurement techniques to OSS products. This contrasts with the commercial case, where code is generally not available for evaluation.

3.2 Community

Contrary to what happens in the product domain, development communities are the main differencing aspect between commercial and OSS products. Not only that development happens in a loose community of (often volunteer) peer developers with almost no hierarchy, also many important assets of the community are open for inspection. This way, mailing lists, discussion forums, version management repositories, bug tracking systems, and a number of other resources are available on the Internet for interested parties to study and contribute to.

Our approach to OSS community evaluation is based on looking at such open community assets in order to assess relevant community quality characteristics. Our base assumption regarding community quality is twofold. On the one hand, certain characteristics of the community strongly influence product quality, especially when observed over an extended period of time. On the other hand, the ability of an OSS

Table 4. Summary of QualOSS product quality characteristics

Characteristic	Definition
Maintainability	The degree to which the software product can be modified. Modifications may include corrections, improvements or adaptation of the software to changes in environment, and in requirements and functional specifications.
Reliability	The degree to which the software product can maintain a specified level of performance when used under specified conditions.
Transferability (Portability)	The degree to which the software product can be transferred from one environment to another.
Operability	The degree to which the software product can be understood, learned, used and is attractive to the user, when used under specified conditions.
Performance	The degree to which the software product provides appropriate performance, relative to the amount of resources used, under stated conditions.
Functional Suitability	The degree to which the software product provides functions that meet stated and implied needs when the software is used under specified conditions.
Security	The ability of system items to protect themselves from accidental or malicious access, use, modification, destruction, or disclosure.
Compatibility	The ability of two or more systems or components to exchange information and/or to perform their required functions while sharing the same hardware or software environment.

Table 5. Summary of QualOSS community quality characteristics

Characteristic	Definition
Maintenance capacity	The ability of a community to provide the resources necessary for maintaining its product(s) (e.g., implement changes, remove bugs, provide support) over a certain period of time
Sustainability	The likelihood that an OSS community remains able to maintain the product or products it develops over an extended period of time.
Process Maturity	The ability of a developer community to consistently achieve development related goals (e.g., quality goals) by following established processes. Additionally, the level to which the processes followed by a development community are able to guarantee that certain desired product characteristics will be present in the product.

community to remain active over time is obviously very important for product survival, and thus very relevant when considering sustainability. Table 5 summarizes the main QualOSS quality characteristics for community assessment.

In the following sections, we discuss these characteristics in more detail, and provide information on how we evaluate them.

Maintenance Capacity
Two aspects are particularly relevant to maintenance capacity; namely, the number of contributors, and the amount of time they are able and willing to contribute to the development effort. The combination of these two aspects basically determines the amount of effort available to maintain the product at a given point in time. If this amount of effort is too small in relation to the actual product size, there is a high risk that important aspects of product maintenance will be neglected. Additionally, a community requires members with the ability and willingness to perform all required maintenance tasks (e.g., programming, testing, documenting, bug triaging, managing releases, etc.)

Due to the difficulty of contacting members directly in a reliable way, measuring the size and time availability of an OSS community is difficult to do directly. However, valuable information can be derived by looking at certain community assets. For instance, an analysis of a project's version repository provides data such as the number of past and present contributors, and the size and frequency of their contributions. When this data is combined with data coming from the analysis of mailing list, forum, instant messaging (e.g., IRC), or bug tracking system archives, a general profile of the contributor community and its level of activity can be produced. Maintenance capacity can be assessed by comparing this profile with the actual product size.

Sustainability
In addition to the factors mentioned for maintenance capacity, sustainability is affected by additional factors, such as composition of the community, and its regeneration ability. Regarding composition, for example, if a community is mainly composed of employees of a particular company, there is a high risk of the project disappearing if the company has financial problems or moves its business focus to other projects. On the other hand, a community that is composed only of volunteers may be less likely to disappear suddenly, but may also have less resources available to keep the project running over time. Consequently, heterogeneity in an OSS community is expected to enhance sustainability.

With respect to regeneration ability—that is, the capacity of a community to recover from member retirement or temporal inactivity by engaging new members—aspects such as the attitude and values of a community and whether its project is considered interesting and technically challenging by potential contributors, play a central role.

Once again, community assets contain a good deal of data that can be useful to evaluate the previous aspects. For example, by looking at the contributors' email or web addresses, it is possible to guess their affiliations. Even if such an approach is not likely to ever become completely reliable, it can provide a good measure of a community's heterogeneity. With regard to regeneration ability, a community's past regeneration can be a good predictor of its future behavior in this respect. Past regeneration can be observed, for example, by analyzing contributions to the version management system or the mailing lists over time.

3.3 Process Maturity Measurement

We believe that the idea of assessing an OSS community in order to determine which good practices it follows and how established these practices are, is perfectly reasonable. Still, as already stated, it is true that existing process assessment models (e.g., CMMI-DEV [1] or SPICE [2]) cannot usually be applied directly to OSS, as they include too many elements that are specific to companies and other conventional development organizations.

Although most, if not all, OSS communities are still quite far from reaching the levels of process discipline expected by the higher levels of standards such as CMMI-DEV, there is evidence of good practices being applied in an established and disciplined fashion by OSS communities. We think that many of these practices correspond to the spirit, if not directly to the letter, of the practices and goals specified by common process maturity standards.

Some examples of disciplined practices observed in prominent OSS communities are:

- *Version/Configuration Management*: Many OSS projects rely on advanced versioning tools to manage their source code. In most cases, access to such systems will be carefully regulated, and the processes for creating new versions are well established and enforced.
- *Release Management*: The GNOME Desktop environment project, as well as the popular GNU/Linux distribution Ubuntu, both have strict 6-month release cycles that have been successfully operating for years. The complex coordination process required for each such cycle is well documented and carefully supervised and enforced by an established release board.
- *Requirements Analysis*: The community behind the Python programming language has a well-documented requirement elicitation and management process as represented by the Python Improvement Proposals (PIPs). Proposals for language enhancements are presented by community members and thoroughly refined until they are considered ready for implementation. The process is conducted in the open and actively enforced by the community.

Similarly to standard maturity assessment approaches, determining the existence of such good practices is mainly a manual procedure. This procedure can rely on information sources such as mailing lists, forum discussions, and published procedures. In addition, indirect evidence can be obtained by looking for the effects of certain practices (e.g., checking whether patch submission procedures are enforced by the community).

Currently, we are in the process of adapting CMMI-DEV and SPICE to the specifics of OSS in order to provide a framework to organize and rate the observations, as well as provide an overall indicator for process maturity. The resulting model shall provide a set of externally observable criteria that a careful, independent observer can objectively assess by looking at the practices followed by a community. This should result in a measure, both quantitative and qualitative, of the process maturity of said community.

3.4 Aggregation and Interpretation

One of the main goals of the QualOSS project is to provide potential OSS users with a means to determine the quality of available products (and of the projects that develop them). Above, we have introduced the QualOSS model framework with a hierarchy of quality characteristics and metrics. When evaluated on OSS products and projects, the metrics provide raw data. In order to have a high-level assessment of the quality of the OSS project, however, this raw, low-level data must be analyzed, interpreted and aggregated to provide a clear view that is easy to understand.

The QualOSS model uses indicators to create the connection between quality characteristics and their metrics (see Fig. 3). Indicators interpret a set of metrics with respect to a specific quality attribute; that is, an indicator consolidates them into a single value that can be used by decision makers to assess a product's quality regarding the corresponding attribute. While *indicators* define interpretations for "leaf" quality attributes, interpretation for higher-level quality attributes is derived through *aggregation* (or abstraction) of the lower-level attributes.

Because the interpretation needs to be easy to understand, interpretation values should have a form that is easy for general decision makers and users (as opposed to software quality experts) to understand and use. In the QualOSS prototype model, all interpretations (for indicators as well as abstractions) can take four different values on a nominal scale (black/red/yellow/green; see [13]), which are ordered (worst to best) as follows: black, red, yellow, and green. To facilitate interpretation, an underlying interval scale (ranging from -100 to +100) is added to the interpretation. The meaning of the different interpretations is defined as follows:

Green: No or minor change of object required to achieve sufficient quality.
Yellow: Significant rework needed.
Red: Critical, needs serious rework.
Black: Rework does not pay off; better "discard" the product, start from scratch.

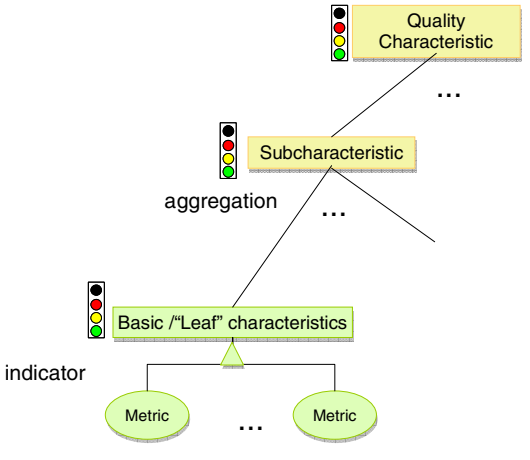


Fig. 3. The QualOSS interpretation model

For indicators, interpretation rules have to be specified to map a set of metrics onto this scale, while aggregation is based on aggregation rules. There are two basic approaches to specify these rules: Calculating a weighted average of underlying (normalized) metrics or indicators, or by specifying logic formulas (such as "if at least one of the lower-level interpretations is red, then the status of the higher-level characteristic is red").

The definition of interpretation and aggregation rules or formulas can be based on expert estimation. If sufficient data are available, experts should rely on analyzing it (e.g., mean values/quartiles) to define aggregation rules. In many cases, though, the amount of data will not suffice for quantitative analysis.

4 Evaluations and Results

This section describes goals and approach of the evaluation of the QualOSS prototype model as well as its results.

4.1 Evaluation Goals

In order to achieve this purpose, the initial plan for validating the QualOSS model [14] defined three potential evaluation goals (EG):

- **EG1:** Evaluate the definition of the quality model (i.e., the quality characteristic definition and prioritization) with the stakeholders.
- **EG2:** Evaluate the usefulness and usability of the QualOSS model. This goal addresses the question to which degree the user believes that the QualOSS provides support for an effective evaluation of the OSS components and to which degree using the QualOSS model/tool is free of effort.
- **EG3:** Evaluate the validity and reliability (accuracy) of QualOSS model; that is, the degree to which the results of the QualOSS evaluation reflect the users' intuition and perception of OSS components.

In this paper, we report on the evaluation of EG1 [12]. That is, the goal of the evaluation reported in this section was to validate the definition and prioritization of the quality (sub)characteristics of the QualOSS model, compared to the perception and intuition of the stakeholders (i.e., evaluators of OSS components). More precisely, this means to assess the QualOSS model and its (sub)characteristics with respect to three aspects:

- **Understandability:** Are the defined quality characteristics understandable/meaningful to the users?
- **Completeness:** Are relevant characteristics missing from the user's point of view?
- **Relevance:** Are the defined quality characteristics relevant for the evaluation of OSS components from the user's perspective?

4.2 Evaluation Approach and Execution

To evaluate the QualOSS model's definitions, we chose to develop and conduct structured interviews, because they allow to clarify and to delimit open-ended

questions. For each of the quality characteristics and sub-characteristics, we stepped through a set of questions aimed at rating the *understandability* of the item on a four-point Likert scale (completely meaningful, … completely meaningless) and rating its *relevance* on a semantic differential scale (0 .. 10). To assess *completeness*, we asked open questions about whether any issues were missing from the interviewee's viewpoint. Table 6 gives an overview of the questionnaire's structure.

We interviewed six industrial partners, most of whom were responsible for IT in their organizations at the time they were interviewed. The sample included four different domains: Public administration, E-government, Research Centre, and software development for the public sector.

Table 6. Number of quality characteristics covered by the survey

Main quality characteristics		# Characteristics	#Sub-characteristics	# Questions
Evolva-bility	Product Evolvability	6	16	72
	Community evolvability	5	13	59
Robustness	Product Robustness	4	10	46
	Community Robustness	3	6	30
Total		18	45	207

4.3 Evaluation Results

The results of the evaluation can be summarized as follows; for a detailed insight, please refer to the QualOSS deliverable D1.6 [14]:

- *Understandability:* The respondents judged the QualOSS model to be sufficiently understandable (concretely, 14 out of 19 evaluated characteristics achieved an average rating of 70% or higher; for subcharacteristics, the results were similar). However, they pointed out specific ambiguities in some quality characteristic definitions.
- *Relevance:* Some items were rated as having low relevance (concretely, 3 out of 15 evaluated characteristics; for subcharacteristics, the results were similar). However, most characteristics of the model were considered highly important (i.e. they achieved a score of 70% or higher).
- *Completeness:* The respondents made six suggestions for refinement of characteristics. That is, from the interviewed stakeholders' viewpoint, only few relevant aspects have not yet been included in the QualOSS model.

It is important to note that, because of the limited number of interviews conducted and the use of a convenience sample, the results described above are neither conclusive nor complete. Nevertheless, they represent a starting point for improving the definition of the QualOSS model.

Finally, regarding the relevance of the quality characteristics, the results do not show a significant difference between the quality characteristics or the granularity

level of the OSS component. The reason for this may be that a larger sample is necessary for identifying significant differences.

5 Summary / Future Work

The main benefit of a comprehensive OSS assessment approach, as presented in this paper, will be a better foundation to judge a community's ability to deliver high-quality software, as well as its long-term sustainability ("will this project exist in 10 years?"). Contrary to many beliefs, sustainability of suppliers is critical to many stakeholders, and is also a problem with commercial software. For example, the European defense consortium EADS-Astrium decided to turn a critical piece of software into OSS in order to become independent of specific suppliers (see, for example, http://cps.erp5.org/).

Moreover, highly regulated industries, such as the automotive, medical, or pharmaceutical industries, have established standards for evaluating software [4, 5]. One consequence of strict process regulations is that software acquisition also commonly requires a thorough assessment of the acquired software components, as well as of the suppliers providing them. When the component in question is an OSS component, the lack of a traditional software provider becomes a serious hindrance, since standard criteria used to evaluate COTS suppliers can generally not be applied directly to OSS.

The goal of our work is to create a framework for comprehensive OSS assessment that is not restricted to the OSS product alone but also addresses community (i.e., supplier) and process maturity aspects. We expect this framework to support decision makers in evaluating OSS, and ultimately facilitate adoption of OSS products in industry.

Next steps in our work include updating and improving the presented prototype model, based on the evaluation results, and evaluation of the improved model in several industrial case studies.

Acknowledgments. This work was supported in part by the EU QualOSS project (grant number: 033547, IST-2005-2.5.5).

References

1. Software Engineering Institute (SEI): Capability Maturity Model Integration (CMMI) for Development, Version 1.2 (2006)
2. ISO/IEC 15504-5:2006, Software Process Improvement and Capability Determination, Part 5
3. Michlmayr, M.: Software Process Maturity and the Success of Free Software Projects. In: Zieliński, K., Szmuc, T. (eds.) Software Engineering: Evolution and Emerging Technologies
4. International Society for Pharmaceutical Engineering (ISPE): Good Automated Manufacturing Practice (GAMP-4) Supplier Guide for Validation of Automated Systems in Pharmaceutical Manufacture (1995)

5. ISO/IEC 61508:1998, Functional safety of electrical/electronic/programmable electronic safety-related systems
6. Software Engineering Institute (SEI): CMMI for acquisition, Technical Report DMU/SEI-2007-TR-017 (2007)
7. Method for Qualification and Selection of Open Source software (QSOS) version 1.6 ©, Atos Origin (April 2006), http://qsos.org/
8. OpenBRR.org, Business Readiness Rating for Open Source©: A Proposed Open Standard to Facilitate Assessment and Adoption of Open Source Software, BRR (2005), http://www.openbrr.org
9. Golden, B.: Open Source Maturity Model © Navica, http://www.navicasoft.com/pages/osmmoverview.htm
10. Duijnhouwer, F., Widdows, C.: Open Source Maturity Model, Capgemini Expert Letter (August 2003), http://SeriouslyOpen.org
11. Deprez, J.C., Alexandre, S.: Comparing Assessment Methodologies for Free/Open Source Software: OpenBRR and QSOS. In: Jedlitschka, A., Salo, O. (eds.) PROFES 2008. LNCS, vol. 5089, pp. 189–203. Springer, Heidelberg (2008)
12. Ciolkowski, M., Guzmán, L., Soto, M., Kamseu, F.: Validation and Calibration of User Manual, QualOSS Deliverable D1.6 (2008)
13. Ciolkowski, M., Soto, M., Monfils, F.F., García, C., Izquierdo, D., Kamseu, F., del Castillo, A.: Calibration of the Prototype QualOSS Model, QualOSS Deliverable D1.5 (2007)
14. Ciolkowski, M., Soto, M., Deprez, J.-C., Monfils, F.F., Kamseu, F., Ruiz, J., del Castillo, A., Izquierdo, D.: Metrics System and Prototype QualOSS Models. QualOSS Deliverable D1.3 (2007)
15. ISO/IEC 9126 International Standard, Software engineering – Product quality, Part 1: Quality model (2001)

Analysing Bug Prediction Capabilities of Static Code Metrics in Open Source Software

Javed Ferzund, Syed Nadeem Ahsan, and Franz Wotawa

Institute for Software Technology, Technische Universität Graz, Austria
{jferzund,sahsan,wotawa}@ist.tugraz.at
http://www.ist.tugraz.at

Abstract. Open Source Softwares provide a rich resource of empirical research in software engineering. Static code metrics are a good indicator of software quality and maintainability. In this work we have tried to answer the question whether bug predictors obtained from one project can be applied to a different project with reasonable accuracy. Two open source projects Firefox and Apache HTTP Server (AHS) are used for this study. Static code metrics are calculated for both projects using in-house software and the bug information is obtained from bug databases of these projects. The source code files are classified as clean or buggy using the Decision tree classifier. The classifier is trained on metrics and bug data of Firefox and tested on Apache HTTP Server and vice versa. The results obtained vary with different releases of these projects and can be as good as 92 % of the files correctly classified and as poor as 68 % of the files correctly classified by the trained classifier.

Keywords: Bug predictor, static code metrics, open source software, empirical software engineering.

1 Introduction

A large part of time and resources is consumed in software testing during the development of a software. We can save this effort if we can find the parts of the source code where the probability of bugs is more and apply these resources on files which require it most. A lot of work have been done to predict bugs in a variety of ways. Most of the studies indicate a relationship between the metrics and number of bugs like [11] and [15]. The research relies on software configuration management systems and bug databases hold important information related to bugs and changes made to files. Other approaches are mainly used to predict the number of bugs [9].

In order to predict the number of bugs or to provide a predictor with regard to a classification schema there are two approaches possible. The first approach uses statistical methods like multiple linear regression, logistic regression, and principal components analysis [13]. Linear regression can be successfully used if the dependent variables change linear with the independent variables. As most of the metrics normally correlate with each other, there is a strong need to

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 331–343, 2008.

overcome the multicolinearity problem. Principal component analysis is used in this respect to reduce the multicolinearity effect. Logistic regression can be used for binary classifications.

The second approach relies on machine learning techniques like decision tree induction, support vector machine, artificial neural networks, k-nearest neighbors to mention some of them. Machine learning techniques have the ability to learn from past data and these techniques can be employed in a variety of complex situations (see [18]).

In this paper, we make use of Decision Tree classifier in order to provide a predictor to classify source files as buggy or clean. We automatically obtain a classifier from one version of a project and test it on different versions of other project and vice versa. We obtained the used bug information from the concurrent version system (CVS), Subversion (SVN) and Bugzilla. We calculate the metrics for each file using our own software. The obtained classifiers correctly classify from 68% to 92% of the files for different releases.

The paper is organized as follows. We first discuss related research. Afterwards, we introduce our approach and present the obtained empirical results. Finally, we conclude the paper.

2 Related Work

A lot of research has been carried out regarding complexity metrics and prediction of bugs. Chidamber and Kimerer initially proposed the object-oriented metrics[8]. Gyimothy et al. validated the object-oriented metrics for fault prediction in open source software [11]. They used logistic regression and machine learning techniques to identify faulty classes in mozilla. Ostrand et al. used code of the file in current release and fault and modification history of the previous releases to predict the expected number of faults in each file of the next release[15]. Ahsan et al. provided a way to analyse program change patterns using the data from version control systems and bug tracking systems [6].

Porter and Selby [16] used classification trees based on metrics from previous releases to identify components having high-risk properties. The authors developed a method of automatically generating measurement-based models of high-risk components. Aljahdali et al. [7] used feed-forward neural network to predict the faults in a program during the initial test/debug process. They also compared the results between regression models and neural networks. Neumann used principal component analysis and artificial neural networks for software risk categorisation [14]. He provided a technique with the capability to discriminate data sets that include disproportionately large number of high-risk software modules. Fenton and Neil have provided a critical review of the defect prediction models based on software metrics [9]. They have discussed the weaknesses of the models proposed previously and identified causes for these shortcomings.

Ferzund et al. [10] applied machine learning techniques to classify faults into different levels. They classified files as low buggy, medium buggy and highly buggy based on the number of faults. Venkata et al. [17] applied different machine learning techniques to develop a predictor model. They used models on real time software defect data and concluded that a combination of 1R and Instance-Based learning along with consistency based subset evolution techniques provides better results as compared to other techniques. Koru et al. [12] combined static software measure with defect data at class level and applied different machine learning techniques to develop bug predictor model.

3 Study Approach

In this section we describe the steps followed to obtain the results. We first describe the data extraction process and the related tasks. Then, we describe the static code metrics used in this study and the calculation for these metrics. Finally, we discuss how to obtain the bug prediction model and related concepts.

3.1 Data Extraction

We checked out source code of Apache HTTP Server (AHS) version 1.3.x and 2.0.x using Subversion (SVN) client. Subversion is a software versioning system used to maintain versions of files including source code and documentation [3]. A collection of bug reports related to AHS was obtained from Bugzilla. To find the files in which these bugs were fixed, we also obtained log information from SVN repositories. The SVN log records revision number, author, date and time, lines modified and the comment by the developer. If a problem was fixed, the developer also mentions the Problem Report number. We filtered the comments for the occurrence of bugs using the regular expression e.g. if a comment contained the keyword fix or Fix or PR: or Fixed, we marked that revision as buggy. To assign number of bugs to individual files, we selected two dates for each version and counted the number of bugs that were fixed between these two dates.

- For AHS 1.3.x 15-08-2002 to 15-08-2005
- For AHS 2.0.x 01-12-2003 to 31-12-2005

We downloaded source code of Firefox Releases 0.8, 1.0, 1.5 and 2.0 using the ftp server. Bug information related to Firefox was obtained from Bugzilla [2]. To find the files in which these bugs were fixed, we also obtained log information from CVS repositories [1]. We processed the comments in CVS log output to find the revisions of files in which bugs were fixed. Each file was assigned a bug count indicating the number of bugs that occurred between two consecutive releases of Firefox. As AHS source code mainly consists of C files and Firefox source code contains C, C++ and JAVA files as well, we selected C files from both projects. We processed the following number of C files in different releases of both projects:

Project	Release	No. of C Files
AHS	1.3.x	155
AHS	2.0.x	191
Firefox	0.8	1389
Firefox	1.0	1387
Firefox	1.5	1522
Firefox	2.0	1527

AHS releases contain more buggy files than Firefox releases. AHS 2.0.x has a highest %age of buggy files with 92% buggy files followed by AHS 1.3.x having 76% buggy files. Firefox 0.8 and 2.0 have the lowest %age of buggy files with less than 2% buggy files. Figure 1 compares the buggy files in all releases of both projects.

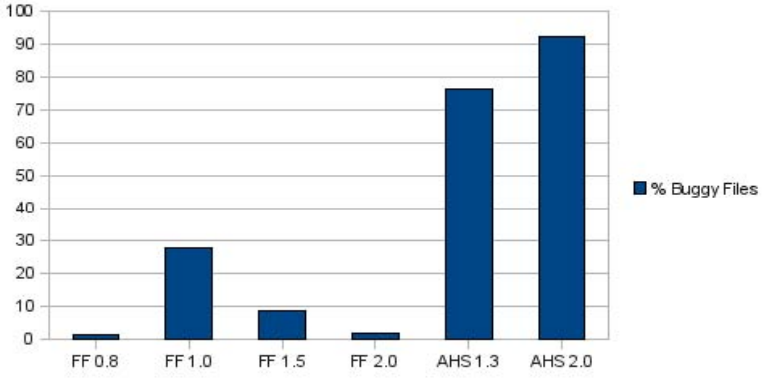


Fig. 1. %age of Buggy Files

3.2 Metrics Calculation

We used our own software to calculate the static code metrics. As C is a procedural language, we calculated individual metrics for functions and aggregated these metrics on files getting total and maximum values of each metrics for each file. We used the following function metrics for our study:

- *NEL(Number of Executable Lines).* The NEL is number of lines of code in a function excluding the blank and comment lines.
- *CD(Control Density).* The CD is the ratio of control lines and executable lines in a function.
- *CC(Cyclomatic Complexity).* The CC is the number of linearly independent paths through a function.
- *PC(Parameter Count).* The PC is the number of arguments a function receives.
- *RP(Return Points).* The RP is the number of return statements in a function.

– *LVC(Local Variable Count).* The LVC is the number of variables locally defined in a function.
– *ND(Nesting Depth).* The ND is the maximum depth of the nested scope in a function.

We also calculated following file metrics in addition to the aggregated metrics:

– *LOC(Lines Of Code).* The LOC is the total number of lines in a file.
– *NOF(Number Of Functions).* The NOF is the total number of functions in a file.
– *NOIF(Number Of Included Files).* The NOIF is the number of files included in a file using the include statement.
– *NOGC(Number Of Global Conditions).* The NOGC is the number of conditional statemnts globally defined in a file.

Table 1 and 2 show the maximum, minimum and average values for the function metrics discussed above. Table 3 shows the maximum, minimum and average values for the file metrics discussed above.

If we compare both projects, the average and minimum values for the metrics are almost similar however there are differences in maximum values. In Firefox 14% of the functions have more than 50 executable lines whereas in AHS 16% of the functions contain more than 50 executable lines. Functions having cyclomatic complexity greater than 20 are 4% in Firefox and 5% in AHS respectively whereas

Table 1. Function Metrics of AHS files

Metrics	AHS 1.3.x			AHS 2.0.x		
	Max	Min	Avg	Max	Min	Avg
NEL	642	2	29	744	2	31
CD	43	0	12	60	0	11
CC	143	1	6	154	1	6
PC	21	0	2	21	0	2
RP	92	0	2	56	0	2
LVC	74	0	2	79	0	3
ND	86	1	2	52	1	3

Table 2. Function Metrics of Firefox files

Metrics	Firefox 0.8			Firefox 1.0			Firefox 1.5			Firefox 2.0		
	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg
NEL	7616	2	29	1931	2	29	12026	2	29	1759	2	30
CD	58	0	11	58	0	11	85	0	11	100	0	11
CC	401	1	5	398	1	5	508	1	5	583	1	6
PC	22	0	2	22	0	2	22	0	2	22	0	2
RP	206	0	2	90	0	2	249	0	2	276	0	2
LVC	191	0	3	191	0	3	191	0	3	282	0	3
ND	302	1	2	302	1	2	302	1	2	302	1	2

Table 3. File Metrics

Project	LOC			NOF			NOIF			NOGC		
	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg	Max	Min	Avg
AHS 1.3.x	4981	7	589	138	1	13	38	0	6	87	0	3
AHS 2.0.x	4801	29	734	111	1	15	34	0	9	43	0	3
Firefox 0.8	9301	13	596	284	1	13	39	0	5	372	0	3
Firefox 1.0	9353	13	592	291	1	13	39	0	5	372	0	3
Firefox 1.5	12677	13	639	312	1	14	40	0	5	372	0	3
Firefox 2.0	9338	13	641	220	1	14	39	0	5	372	0	3

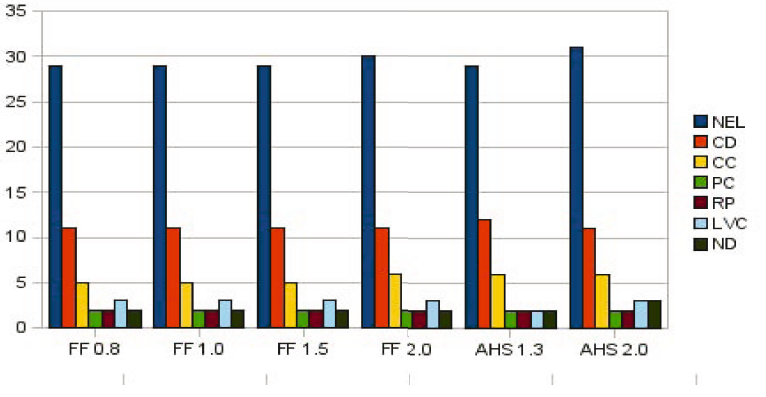


Fig. 2. Average Function Metrics

functions having parameter counts greater than 5 are 4.5% in Firefox and 3% in AHS respectively. Firefox and AHS contain 4.2% and 8.5% of functions having more than 5 return points respectively. 1.8% of the functions in Firefox and 4% of the functions in AHS have more than 10 nesting depth. Figure 2 shows the average values of function metrics for both projects.. Average file sizes are comparable in both projects with slight differences. AHS 2.0 has the highest average file size and AHS 1.3 has the smallest average file size. 15% of files in Firefox 0.8 and Firefox 1.0, 17% of files in Firefox 1.5 and Firefox 2.0, 18% of files in AHS 1.3 and 27% of files in AHS 2.0 have more than 1000 LOC. Figure 3 compares the average file sizes in both projects. Average number of functions/file is similar in both projects. However, 18% of files in Firefox 0.8 and Firefox 1.0, 19% of files in Firefox 1.5 and Firefox 2.0, 16% of files in AHS 1.3 and 48% of files in AHS 2.0 contain more than 20 functions. In Firefox 5% files contain more than 10 globally defined conditions whereas in AHS 6% files contain more than 10 globally defined conditions. Firefox contains 14 % files having more than 10 files included using the #include statement whereas in AHS 1.3 and AHS 2.0 this amount is 12% and 34% respectively. This indicates AHS 2.0 has highly correlated files. Figure 4 compares the file metrics of both projects.

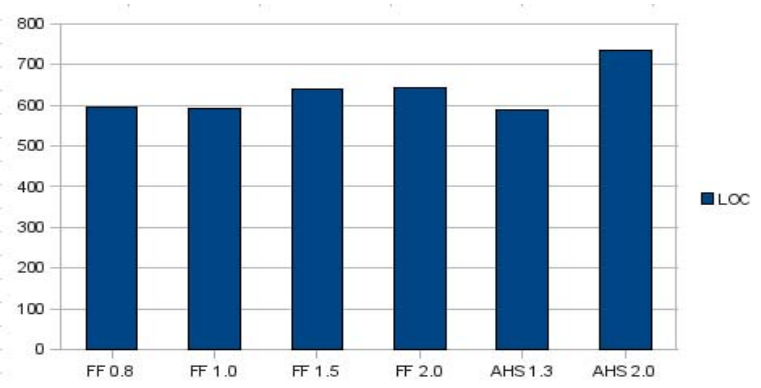


Fig. 3. Average File Sizes

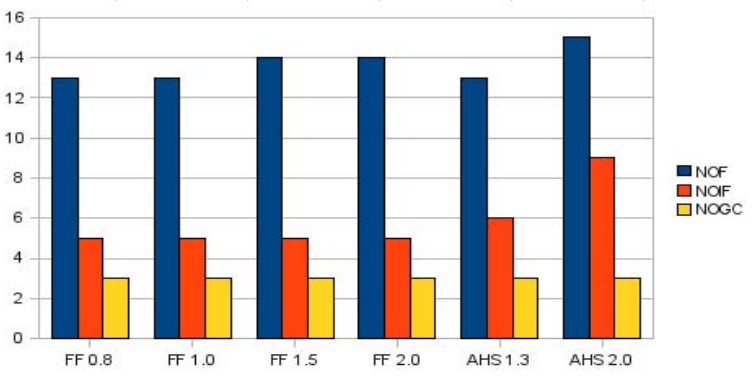


Fig. 4. Average File Metrics

3.3 Obtaining the Bug Predictor

We stored the information obtained from previous two steps into relations. Each relation consisted of files as entities. The relations consisted of file attributes including the static code metrics and the bugs related to each file. We trained a decision tree classifier on this data using WEKA (a Machine Learning tool developed in JAVA)[4] to classify files as clean or buggy. We selected decision tree for our study due to its strong classification capabilities. We trained the classifier on four releases of Firefox and tested each model obtained on two versions of AHS. Alternatively, we also trained the classifier on two releases of AHS and tested each model obtained on four releases of Firefox. After training of classifier WEKA stores models in the form of Java serializable objects holding different options, which later on can be applied to the test data.

A major proportion of time requied to obtain a bug predictor is involved in data preparation. Metrics calculation and extraction of bug information requires

```
0 yes 0.982973149967256 yes (8,182,46,146,35,43,13,25,9,18,5,13,4,14,5,11,1,293)
1 yes 0.982973149967256 yes (8,205,78,79,23,38,16,18,4,22,8,26,9,24,11,9,0,304)
2 yes 0.982973149967256 yes (6,122,55,52,20,17,9,13,5,13,5,6,3,8,3,8,0,227)
3 yes 0.982973149967256 yes (9,185,52,79,19,32,11,23,4,22,7,21,8,13,5,10,0,282)
4 yes 0.982973149967256 yes (43,1455,212,544,28,230,32,111,5,108,18,124,12,97,14,19,2,2096)
5 yes 0.982973149967256 yes (3,35,27,0,0,0,0,9,7,2,1,1,1,3,1,2,0,71)
6 yes 0.982973149967256 yes (4,107,61,29,18,17,9,12,5,4,2,10,7,7,3,15,1,181)
7 yes 0.982973149967256 yes (35,871,177,307,35,157,61,82,7,62,7,55,14,55,11,10,1,1139)
8 yes 0.982973149967256 yes (17,1429,488,176,23,192,79,56,5,87,33,88,23,48,8,17,0,1752)
9 yes 0.982973149967256 yes (11,440,138,105,23,73,26,32,5,15,4,34,9,46,22,15,1,540)
10 yes 0.982973149967256 yes (11,241,73,115,23,36,11,27,5,24,6,19,6,13,2,13,4,405)
11 yes 0.982973149967256 yes (28,488,79,216,25,55,7,70,5,36,5,43,9,38,3,6,0,735)
12 yes 0.982973149967256 yes (31,1045,124,282,24,132,17,86,5,79,8,110,14,69,13,9,0,1495)
13 yes 0.982973149967256 yes (6,46,13,7,7,2,2,15,5,5,2,6,4,6,1,5,0,97)
14 yes 0.982973149967256 yes (39,1556,227,435,23,197,25,118,7,115,9,128,16,79,11,11,5,2130)
15 yes 0.982973149967256 yes (8,92,24,72,25,16,6,7,2,9,3,9,3,9,2,6,0,140)
16 yes 0.982973149967256 yes (71,3833,356,877,25,565,51,135,5,343,28,294,20,164,13,12,0,4801)
17 yes 0.982973149967256 yes (18,743,155,254,25,106,22,34,6,26,4,63,20,63,18,6,0,1116)
18 yes 0.982973149967256 yes (2,9,5,0,0,0,0,1,1,1,1,0,0,2,1,4,0,33)
19 yes 0.982973149967256 yes (6,111,75,12,12,16,16,12,5,6,3,6,5,6,1,4,0,194)
20 yes 0.982973149967256 yes (34,1611,489,367,25,207,40,49,7,96,18,95,17,94,15,9,1,2020)
21 yes 0.982973149967256 yes (12,585,100,166,21,76,12,22,5,44,6,46,10,37,7,8,1,791)
22 yes 0.982973149967256 yes (5,47,26,12,12,4,4,12,5,4,2,3,3,6,2,7,0,92)
23 yes 0.982973149967256 yes (7,100,28,45,16,10,3,4,1,8,3,8,2,7,1,7,2,171)
24 yes 0.982973149967256 yes (10,188,83,133,33,26,8,12,4,15,3,11,5,14,3,5,2,290)
25 yes 0.982973149967256 yes (16,195,21,169,37,34,7,33,4,20,5,14,3,17,2,5,3,290)
```

Fig. 5. Predictions for individual files

much time which depends on the project size and it may take hours even using a high speed processor. Once the data is prepared, weka classifiers take few seconds to learn and produce outputs. Weka classifiers hold multiple options to be used during training and testing. Following is a brief summary of these options.

- *-t <name of training file >*. Sets training file.
- *-T <name of test file >*. Sets test file. If missing, a cross-validation will be performed on the training data.
- *-c <class index >*. Sets index of class attribute (default: last).
- *-x <number of folds >*. Sets number of folds for cross-validation (default: 10).
- *-s <random number seed >*. Sets random number seed for cross-validation (default: 1).
- *-l <name of input file >*. Sets model input file.
- *-d <name of output file >*. Sets model output file.
- *-i*. Outputs detailed information-retrieval statistics for each class.
- *-k*. Outputs information-theoretic statistics.
- *-p <attribute range >*. Only outputs predictions for test instances, along with attributes (0 for none).

During training -t and -d options can be used to train on an input file and store the model obtained into a model file which later can be used. For predictions or testing -T and -l options can be used to mention test and model files. For getting output statistics -i and -k options are used. In order to get output predictions for individual instances -p option is used. It displays the actual and predicted

values for each instance. Figure 5 displays a screenshot of the predictions, the values in each line are separated by a single space. The fields are the zero-based test instance id, followed by the predicted class value, the confidence for the prediction (estimated probability of predicted class), the true class and the values of selected attributes [5].

4 Results

In this section we discuss the results obtained by applying the Decision Tree classifier. We obtained the number of correctly classified instances (CCI), the Kappa statistics (KS), the mean absolute error (MAE), and the root mean squared errors (RMSE). KS is a means of classifying agreement in categorical data.

$$K = \frac{P(A)-P(E)}{1-P(E)}$$

where P(A) is the proportion of times the model values are equal to the actual values and P(E) is the proportion of times the model values are expected to agree with actual values by chance. A KS value of 1 means a statistically perfect modeling whereas a 0 means every model value was different from the actual value. MAE is the average of the difference between predicted and actual value in all test cases. It is calculated by taking the sum of absolute values of errors and then dividing by number of predictions.

$$MAE = \frac{|a1-c1|+|a2-c2|+\cdots+|an-cn|}{n}$$

The RMSE measures the average magnitude of the error. It is calculated by taking the average of the squared differences between each computed value and its corresponding correct value.

$$RMSE = \sqrt{\frac{(a1-c1)^2+(a2-c2)^2+\cdots+(an-cn)^2}{n}}$$

Using the predictor obtained from AHS 1.3.x, we correctly classified 92% of the files in Firefox 0.8 and Firefox 2.0. However for Firefox 1.0 the predictor could correctly classify 68% of the files. The results for Firefox 1.5 were inbetween with a value of 84% correctly classified files. The results were almost similar using the predictor obtained from AHS 2.0.x. Table 4 shows that MAE values are below 0.4, indicating fair accuracy of results. In most cases RMSE values are near 0.4 which further validates the results.

Using the predictors obtained from Firefox 0.8 , Firefox 1.5 and Firefox 2.0, we correctly classified 92% of the files in AHS 2.0.x. However for AHS 1.3.x the predictors could classify 78% of the files. The predictor obtained from Firefox 1.0 showed poor results with 78% and 68% of the files of AHS 2.0.x and AHS 1.3.x correctly classified respectively. Table 5 shows that MAE values in most cases are below 0.3 indicating good accuracy of results. The RMSE values are slightly higher in some cases which indicate poor accuracy.

We have also calculated the True Positive rate, False Positive rate, Precision, Recall and F-Measure for the predictions obtained by applying each model. The

Table 4. Classification of Firefox files using the predictor obtained from AHS

Predictor	Release	CCI	KS	MAE	RMSE
AHS 1.3.x	Firefox 0.8	92%	0.0586	0.18	0.28
	Firefox 1.0	68%	-0.036	0.36	0.51
	Firefox 1.5	84%	-0.003	0.24	0.37
	Firefox 2.0	91%	0.097	0.19	0.30
AHS 2.0.x	Firefox 0.8	92%	0.01	0.12	0.29
	Firefox 1.0	70%	0.04	0.31	0.52
	Firefox 1.5	86%	0.02	0.16	0.36
	Firefox 2.0	92%	-0.02	0.12	0.29

Table 5. Classification of AHS files using the predictor obtained from Firefox

Predictor	Release	CCI	KS	MAE	RMSE
Firefox 0.8	AHS 1.3.x	78%	0.0	0.22	0.45
	AHS 2.0.x	92%	0.0	0.09	0.28
Firefox 1.0	AHS 1.3.x	68%	0.0	0.34	0.52
	AHS 2.0.x	78%	0.09	0.29	0.45
Firefox 1.5	AHS 1.3.x	77%	0.04	0.23	0.46
	AHS 2.0.x	92%	0.09	0.12	0.29
Firefox 2.0	AHS 1.3.x	79%	0.0	0.22	0.45
	AHS 2.0.x	92%	0.09	0.09	0.27

Table 6. Using AHS 1.3.x as predictor

Release	TP Rate	FP Rate	Precision	Recall	F-Measure	Class
Firefox 0.8	0.85	0.95	0.98	0.85	0.95	no
	0.05	0.15	0.01	0.05	0.01	yes
Firefox 1.0	0.83	0.91	0.70	0.83	0.76	no
	0.09	0.17	0.17	0.09	0.12	yes
Firefox 1.5	0.88	0.64	0.94	0.88	0.91	no
	0.36	0.12	0.22	0.36	0.27	yes
Firefox 2.0	0.86	0.96	0.98	0.86	0.92	no
	0.04	0.14	0.01	0.04	0.01	yes

True Positive (TP) rate is the proportion of examples which were classified as class x, among all examples which truly have class x, i.e. how much part of the class was captured. It is equivalent to Recall. The False Positive (FP) rate is the proportion of examples which were classified as class x, but belong to a different class, among all examples which are not of class x. The Precision is the proportion of the examples which truly have class x among all those which were classified as class x. The F-Measure is simply 2*Precision*Recall/(Precision+Recall), a combined measure for precision and recall [5]. Tables 6-11 show the values of these measures for each predictor. Class "yes" indicates the buggy files whereas class "no" indicates the clean files. For Firefox predictions precision and recall

Table 7. Using AHS 2.0.x as predictor

Release	TP Rate	FP Rate	Precision	Recall	F-Measure	Class
Firefox 0.8	0.93	0.90	0.98	0.93	0.96	no
	0.1	0.07	0.02	0.1	0.03	yes
Firefox 1.0	0.94	0.91	0.73	0.94	0.82	no
	0.1	0.06	0.37	0.09	0.15	yes
Firefox 1.5	0.94	0.91	0.92	0.94	0.93	no
	0.08	0.06	0.11	0.08	0.09	yes
Firefox 2.0	0.93	1.0	0.98	0.93	0.96	no
	0	0.01	0	0	0	yes

Table 8. Using Firefox 0.8 as predictor

Release	TP Rate	FP Rate	Precision	Recall	F-Measure	Class
AHS 1.3.x	1.0	1.0	0.76	1.0	0.86	yes
	0	0	0	0	0	no
AHS 2.0.x	1	1	0.92	1	0.96	yes
	0	0	0	0	0	no

Table 9. Using Firefox 1.0 as predictor

Release	TP Rate	FP Rate	Precision	Recall	F-Measure	Class
AHS 1.3.x	0.82	0.81	0.76	0.82	0.79	yes
	0.19	0.18	0.25	0.19	0.21	no
AHS 2.0.x	0.82	0.67	0.93	0.82	0.87	yes
	0.33	0.18	0.13	0.33	0.19	no

Table 10. Using Firefox 1.5 as predictor

Release	TP Rate	FP Rate	Precision	Recall	F-Measure	Class
AHS 1.3.x	0.97	1	0.76	0.97	0.85	yes
	0	0.02	0	0	0	no
AHS 2.0.x	0.99	0.93	0.93	0.99	0.96	yes
	0.07	0.01	0.33	0.07	0.11	no

Table 11. Using Firefox 2.0 as predictor

Release	TP Rate	FP Rate	Precision	Recall	F-Measure	Class
AHS 1.3.x	1	1	0.76	1	0.86	yes
	0	0	0	0	0	no
AHS 2.0.x	1	1	0.92	1	0.96	yes
	0	0	0	0	0	no

is high for clean file predictions but its poor for buggy file predictions. However for AHS predictions precision and recall is high for buggy file predictions and its poor for clean file predictions. The reason may be the high percentage of buggy files in AHS and high percentage of clean files in Firefox. In AHS more than 70% files are buggy whereas in Firefox less than 10% files are buggy except Firefox 1.0 in which 28% files are buggy, as depicted in Figure 1.

5 Conclusions

The results have shown that predictions vary with different releases however bug predictor obtained from one project can be applied to a different project with a reasonable accuracy. Factors other than static code metrics can be considered for more accurate predictions. Although the average metrics values are similar among releases of Firefox, the occurrence of bugs was high in Firefox 1.0 which causes differences in precision of results. AHS 2.0 holds the highest percentage of buggy files which may be due to large file sizes and high interdependency among files. AHS 2.0 files are larger than other releases in both projects and it contains 27% files having more than 1000 LOC, 48% files having more than 20 functions/file and 34% files having more than 10 files included. AHS 1.3 files have average metrics similar to Firefox but it contains higher percentage of buggy files. It indicates that factors other than static code metrics are also important in inducing bugs like chan! ges made to code, refactorings and developer experties etc.

In the future, we want to use process metrics along with static code metrics to find a set of metrics which can be used for prediction of bugs in a variety of projects. We also want to increase the number and size of projects for this study.

Acknowledgments. This research presented in this paper is partly funded by Higher Education Commission(HEC), Pakistan under its scholarship programme and partially conducted within the competence network Softnet Austria (www.soft-net.at) that is funded by the Austrian Federal Ministry of Economics (bm:wa), the province of Styria, the Steirische Wirtschaftsfrderungsgesellschaft mbH. (SFG), and the city of Vienna in terms of the center for innovation and technology (ZIT).

References

1. anonymous@cvs-mirror.mozilla.org
2. http://www.bugzilla.mozilla.org/
3. http://subversion.tigris.org/
4. http://www.cs.waikato.ac.nz/ml/weka/
5. http://weka.sourceforge.net/wekadoc
6. Ahsan, S.N., Ferzund, J., Wotawa, F.: A Database for the Analysis of Program Change Patterns. In: Proceedings of the 4th International Conference on Networked Computing and Advanced Information Management, Gyeongju, Korea, September 2-4 (2008)

7. Aljahdali, S.H., Sheta, A., Rine, D.: Prediction of software reliability: a comparison between regression and neural network non-parametric models. In: ACS/IEEE International Conference on Computer Systems and Applications, June 25-29, 2001, pp. 470–473 (2001)
8. Chidamber, S.R., Kemerer, C.F.: A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering 20(6), 476–493 (1994)
9. Fenton, N., Neil, M.: A Critique of Software Defect Prediction Models. IEEE Transactions on Software Engineering 25(5) (September 1999)
10. Ferzund, J., Ahsan, S.N., Wotawa, F.: Automated Classification of Faults in Programms using Machine Learning Techniques. In: Artificial Intelligence Techniques in Software Engineering Workshop, July 21 (2008)
11. Gyimothy, T., Ferenc, R., Siket, I.: Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction. IEEE Trans. Software Eng. 31(10), 897–910 (2005)
12. Koru, A.G., Liu, H.: Building effective defect-prediction models in practice. Software, IEEE 22(6), 23–29 (2005)
13. Nagappan, N., Ball, T., Zeller, A.: Mining Metrics to Predict Component Failures. In: Proceedings of the 28th international conference on Software engineering, Shanghai, China (November 2005)
14. Neumann, D.E.: An Enhanced Neural Network Technique for Software Risk Analysis. IEEE Transactions on Software Engineering (September 2002)
15. Ostrand, T.J., Weyuker, E.J., Bell, R.M.: Predicting the Location and Number of Faults in Large Software Systems. IEEE Trans. Software Eng. 31(4), 340–355 (2005)
16. Porter, A., Selby, R.: Empirically-guided software development using metric-based classification trees. IEEE Software 7, 46–54 (1990)
17. Venkata, U.B., Challagulla, B., Bastani Farokh, B., I-Ling, Y.: Empirical Assessment of machine Learning based Software Defect Prediction Techniques. In: Proceedings of the 10th IEEE International Workshop on Object Oriented Real- Time Dependable Systems (WORDS 2005). IEEE, Los Alamitos (2005)
18. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco (2005)

Measuring the Impact of Different Categories of Software Evolution

Francesca Longo, Roberto Tiella, Paolo Tonella, and Adolfo Villafiorita

Fondazione Bruno Kessler IRST
via Sommarive, 18
38100 Trento
{longo,tiella,tonella,adolfo}@fbk.eu

Abstract. Software evolution involves different categories of interventions, having variable impact on the code. Knowledge about the expected impact of an intervention is fundamental for project planning and resource allocation. Moreover, deviations from the expected impact may hint for areas of the system having a poor design. In this paper, we investigate the relationship between evolution categories and impacted code by means of a set of metrics computed over time for a subject system.

1 Introduction

During software evolution, implementation of a change request may require a variable amount of resources, depending on the impact of the change. Small, local changes can be accommodated quickly, while changes that affect a high number of modules need careful implementation and regression testing. Knowledge about the expected impact of a change is important for project planning, release scheduling, and resource allocation.

In this paper we investigate the impact of different categories of software evolution interventions. We can classify software evolution into:

1. corrective evolution,
2. code improvement and adaptation (refactoring [3])
3. addition of new functionalities

(see [5] for a similar classification of software maintenance interventions). In this work, we consider points 1 and 2, trying to address the following research questions:

1. **RQ1:** What is the extent of impact of corrective evolution?
2. **RQ2:** What are the effects of refactoring on the code size and organization?

Refactorings [3] cover a wide spectrum of interventions on the code, having in common the improvement of internal properties without affecting the externally visible functionalities. To better analyze the different types of refactorings, we split them into three categories:

1. *Conv*: uniform adoption of coding conventions and style;
2. *FrLibEnv*: adoption of new framework, libraries or environments;
3. *IntStruct*: improvement of the internal structure of the code.

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 344–351, 2008.

More specifically, the goal of RQ1 and RQ2 is measuring how a system changes due to a specific corrective evolution intervention (e.g. a bug is fixed) or to a specific refactoring intervention (e.g. a switch to a new framework). This constitutes a first step to identify what are the metrics most influenced by such interventions and, possibly, identify trends in such metrics. Such trends could then be used for planning purposes (e.g. to predict time, cost, and resources needed for a specific intervention) or to get a better understanding on the evolution of a system (e.g. a deviations from the expected behavior occur may suggest a poor system design).

Table 1. Metrics used to assess the impact of software evolution

Grouping	Metric	Description
Size	NCLS	Number of classes (interfaces excluded): total number of classes defined in the system
	NOM	Number of methods: total number of defined, i.e. implemented, methods in the system
	NCSS	Number of Non Commenting Source Statements: number of lines used in statements and declarations as defined by JavaNCSS project [3]
Abstractness/Inheritance	NABS	Number of abstract classes: total number of classes declared abstract
	NRC	Number of root classes, i.e. classes that derives from JRE's classes or from external libraries
	NII	Number of interfaces implemented: sum over all classes defined in the system of the number of implemented interfaces
	DIT	Depth of Inheritance Tree: sum of the DIT for all class in the system. The DIT for a class is defined as the length of the path to reach the root class (see definition above) walking up the 'extends' relation
	NOC	Number of Children: sum over all classes in the system of the number of direct children in the inheritance tree.
Coupling	CBO	Coupling between Objects: sum over all classes in the system of the CBO as defined in [1]
	RFC	Response For a Class (also counting method invocations of JRE or external classes): sum over all classes in the system of the RFC as defined in [1]
Other	ANOM	Average Number of Methods per class: NOM/NCLS
	ACBO	Average Coupling between Objects per class: CBO/NCLS
	MODAR	Number of Files Modified/Added/Removed

2 Metrics

The set of metrics used to address research questions RQ1 and RQ2 is listed in Table 1.

This set comprises four metrics (NOM, DIT, NOC, CBO) out of the five proposed by Chidamber-Kemerer's [2], some metrics to evaluate the level of code abstractness/inheritance (NABS, NRC, NII), a metric for the code size (NCSS), and a metric to measure the number of modified/added/removed files (MODAR). MODAR is used to evaluate the extent of interventions; Abstractness/Inheritance metrics are used to judge improvements in the structure of the system and evaluating the reuse of code; size-related metrics are used to estimate effort for maintenance and testing.

All but NCSS and MODAR have been collected by means of a tool developed by some of the authors in previous work. NCSS was computed using JavaNCSS [6]. MODAR was computed by analysing CVS logs by means of simple UNIX text utilities (i.e. grep, awk, etc.). The values of the metrics have been computed on given releases of the system, summing values when needed: e.g. NOM is the sum of the all the methods of all the classes defined in the system.

See Table 1 for a brief description of each metric.

3 Case Study

We computed the metrics in Table 1 for the ProVotE system, a Java system developed at FBK-IRST over the last four years.

The ProVotE system is the software controller of DRE+VVPAT e-voting machine (that is, a touch-screen based e-voting machine with voter verified printed audit trail [7,8]). The ProVotE system has been used in Italy with experimental value by about ten thousand citizens and with legal value by about six hundred voters [1].

The ProVotE system architecture is composed of four main components (see Figure 1):

- *services*, that provides the basic functionality to the rest of the application, such as drivers for controlling hardware components, e.g. printer, external 'in-use' indicator, managing logs for audits, and transparently managing redundant and ciphered persistence of data;
- *data model management*, that manages all the election specific data, comprising candidates and parties, the ballot data, per-machine election results, and the symmetric and asymmetric keys used for ciphering and signing;
- *administration*, that provides the poll-workers with a graphical interface to manage the machine during an election;
- *vote*, that provides the elector with a GUI to express her/his vote; the GUI is designed to resemble the traditional paper ballot

Both the administration and the vote components are structured according to the MVC design pattern [4,9] and they comprise a:

– *user interface* component, that manages the graphical layout (view);
– *control logic* component, that defines how the machine has to react to user actions and specifies the logic of the user interface (e.g. what screens has to be shown next) (control).

(The model is implemented by the data management component see Figure 1.)

The latest release of ProVotE consists of 10958 lines of code (255 classes), more or less evenly distributed among the four components of the architecture. Its evolution has been recorded in CVS. Six releases (34 versions) are available for analysis. Bugs have been reported through Mantis.

Bug fixing interventions on ProVotE have been identified by searching for bugs reported in the bug tracking system. A superset of the files modified to fix a bug has been obtained by CVS diff. Then, the differences have been analysed manually in order to determine exactly which files have been impacted by the resolution of the bug. We resorted also to Mantis comments, when available, to perform this task. The output of this analysis is a value for the metric MODAR (described in Table 1).

In order to identify and classify the refactoring interventions, different methods have been adopted: analysis of CVS logs, documentation and project plans (searching for key words like refactoring, restructuring, etc.), developers consultation, and code inspection. From the set of interventions performed on ProVotE over the years, those that were classifiable into the three categories introduced in Section 1 have been selected with the result of a set of 13 interventions (four *Conv*, three *FrLibEnv* and five *IntStruct*).

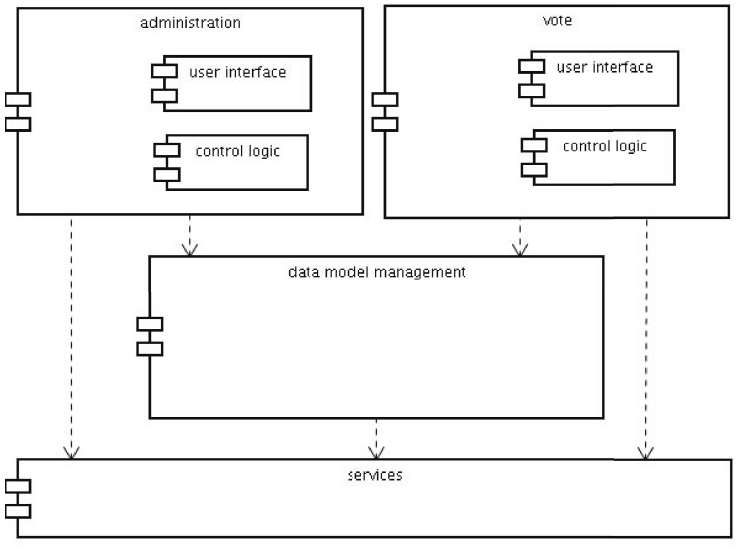


Fig. 1. The ProVotE architecture

4 Results

Table 2 shows the unique identifier of each bug (ID), the priority assigned to each bug (priority), the number of files impacted by the bug fixing intervention (MODAR), the number of total files of the system at the beginning of the intervention (totFiles) and the percentage of modified files with respect to the total number of files (%MODAR). The last row shows the average values of MODAR and %MODAR.

Table 2. Number of files impacted by bug fixing

ID	priority	MODAR	totFiles	%MODAR
330	urgent	2	2244	0.09
230	high	1	2046	0.05
235	high	2	2046	0.05
293	high	2	2227	0.09
300	high	4	2227	0.04
306	high	1	2230	0.04
235	normal	4	2046	0.2
238	normal	2	2046	0.1
240	normal	2	2046	0.1
304	normal	1	2230	0.04
312	normal	1	2230	0.04
231	low	1	2046	0.05
233	low	1	2046	0.05
309	low	3	2230	0.16
311	low	1	2230	0.04
303	none	1	2230	0.18
avg	-	*1.81*	-	*0.08*

Looking at these data it is possible to observe that: (1) at most four files have been modified, corresponding to 0.2% of total files; (2) at least one file has been modified, corresponding to 0.04% of total files; (3) for a corrective intervention 1.81 files on average have been modified, corresponding to 0.08% of total files.

Concerning the results of refactoring analysis, Table 3 shows the values of the metrics presented in Section 2. Data represent percentage delta values, except for ANCSS that represents the absolute value of the variation of lines of code (Absolute NCSS).

5 Discussion

Values regarding corrective evolution show that, for the analysed system, the impact of bug fixing was small. In most cases less than two files have been impacted by a corrective intervention. Another interesting observation is that bugs with high priority do not impact more files with respect to the other cases.

Table 3. Number of files impacted by refactoring

	NCLS	NABS	NRC	NII	NOM	DIT	NOC
Conv1	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Conv2	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Conv2	1.0	0.0	0.0	0.0	0.5	1.0	1.9
Conv4	0.0	0.0	0.0	0.0	-0.2	0.0	0.0
FrLibEnv1	-9.1	26.2	-8.0	-8.5	-14.7	-7.1	-10.0
FrLibEnv2	0.0	100.0	-1.0	50.0	2.9	20.9	0.9
FrLibEnv3	10.0	12.5	5.0	11.5	-5.7	18.4	14.4
IntStruct1	0.5	0.0	1.0	8.3	0.3	0.0	0.0
IntStruct2	0.4	7.7	0.0	0.0	0.2	0.4	0.7
IntStruct3	0.4	0.0	0.9	0.0	0.7	0.0	0.0
IntStruct5	0.8	0.0	0.8	0.0	0.6	0.4	0.7
IntStruct6	0.8	0.0	1.7	0.0	0.8	0.0	0.0

	CBO	RFC	ANOM	ACBO	ANCSS	NCSS	MODAR
Conv1	0.0	-0.4	0.0	0.0	0	0.0	1.2
Conv2	-0.2	0.0	0.0	-0.2	-1	0.0	0.4
Conv3	0.5	0.2	-0.4	-0.5	34	0.4	3.7
Conv4	0.0	0.0	-0.2	0.0	-4	0.0	1.0
FrLibEnv1	-5.8	-8.2	-6.7	3.1	-853	-8.5	25.7
FrLibEnv2	0.7	-12.8	2.9	0.7	-1065	-10.0	36.0
FrLibEnv3	-1.2	-15.7	-14.3	-10.1	-1290	-13.3	41.8
IntStruct1	0.2	0.1	-0.2	-0.3	30	0.3	16.2
IntStruct2	0.9	0.4	-0.1	0.5	43	0.4	0.7
IntStruct3	1.0	-1.0	0.2	0.5	-33	-0.4	28.9
IntStruct5	0.0	0.1	-0.1	-0.8	11	0.1	3.3
IntStruct6	1.7	0.9	0.0	1.0	83	0.8	1.7

This may be interpreted as an indicator of a good system design, capable of limiting the scope of (even high priority) corrective evolution. Correspondingly, the impact of corrective evolution measured by MODAR and %MODAR might be suggested as a project management indicator of design quality, in that it gives an immediate intuition of the locality of corrective changes.

Each category of refactoring interventions needs to be analysed separately. In the case of *Conv*, the impact is small, as expected for this class of code improvement intervention.

Interventions that involve the adoption of new framework or libraries (*FrLibEnv*) have the largest impact. In particular it is possible to notice major changes in all metrics values and in the majority of the cases the NCSS value considerably decreases.

In *FrLibEnv1*, portions of the data-model related code have been removed and replaced by code automatically generated from XSchema descriptions. The impact of such an intervention is quite high (MODAR nearly 26%). This is due to the fact that the model was changed and the client code had to be updated

accordingly. The negative values of most of the metrics are justified by the fact that the hand-written code of the model was highly hierarchical. Replacing it with the automatically generated code resulted in decreasing the values of various metrics among which DIT, NOC, NII.

FrLibEnv2 consists of the extraction of a GUI framework from the existing code of the administration GUI (MODAR impact of 36%). Developers recognized the opportunity to implement part of the user interface as a specific instantiation of a more general architectural pattern. In particular, all the common generalized behaviour related to user interaction, typical of a wizard-based application, was modelled as an abstract layer. Existing classes were adapted to use such an abstraction. Metrics reflects the refactoring activity: abstraction-related metrics increase (NABS, NII, DIT), while the size of the application decreases (NCSS is −10%).

In *FrLibEnv3*, the application logic was modelled with UML statecharts and the corresponding hand-written code replaced by code automatically generated from the UML statecharts. (Code generation is performed using a tool specifically developed for the purpose [10].) A large part of the application was involved in the refactoring (MODAR is about 42%), since the UML statecharts embed information about the flow of screens, the code associated to the navigation among screens (e.g. next and previous buttons), and the controller (e.g. what actions have to be invoked when the user presses a button). NABS, NII, DIT increased as a consequence of having included an interface between the state machine and the delegate classes. NCSS increases as a consequence of having introduced code needed to operate the state machine.

For interventions classified as *IntStruct*, the values of metrics change depending on the portion of code to be improved and the kind of improvement. Common to all the interventions is that the abstractness always increases. In particular, is possible to notice the increase of the number of interfaces (NII) and abstract classes (NABS) in almost all cases.

To summarize, project managers should expect minor impact from corrective evolution (assuming the system has a good design) and coding convention adoption, while a major impact is associated with the adoption of new frameworks and libraries. Improvement of the internal code structure is somewhat in the middle, depending on the specific intervention. In our future work we will investigate this category in more detail.

References

1. Caporusso, L., Buzzi, C., Fele, G., Bertoli, P., Sartori, F.: Transitioning to Electronic Voting and Citizen Participation. In: Krimmer, R. (ed.) Proceedings of Electronic Voting 2006 Conference, pp. 191–200 (2006)
2. Chidamber, S.R., Kemerer, C.F.: A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering 20(6) (June 1994)
3. Fowler, M.: Refactoring: improving the design of existing code. Addison-Wesley, Reading (1999)

4. Gamma, J., Erich, R., Helm, V., Richard, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading (1995)
5. ISO/IEC 14764. Software Engineering Software Life Cycle Processes Maintenance
6. JavaNCSS A Source Measurement Suite for Java, http://www.kclee.de/clemens/java/javancss/
7. Mercuri, R.: Explanation of voter-verified ballot systems. ACM Software Engineering Notes (SIGSOFT) 27(5), http://catless.ncl.ac.uk/Risks/22.17.html
8. Mercuri, R.: A better ballot box? IEEE Spectrum Online (October 2002) http://www.spectrum.ieee.org/WEBONLY/publicfeature/oct02/evot.html
9. Reenskaug, T.: Thing-Model-View Editor an Example from a planning system, Xerox Parc Technical Note (May 1979), http://heim.ifi.uio.no/trygver/1979/mvc-2/1979-12-MVC.pdf
10. Tiella, R., Villafiorita, A., Tomasi, S.: FSMC+, a tool for the generation of Java code from statecharts. In: Proceedings of the 5th International Symposium on Principles and practices of programming in Java (PPPJ 2007), pp. 93–102. ACM, New York (2007)

Using PSU for Early Prediction of COSMIC Size of Functional and Non-functional Requirements

Luigi Buglione[1], Olga Ormandjieva[2], and Maya Daneva[3]

[1] École de Technologie Supérieure (ETS) / Engineering.it
luigi.Buglione@eng.it
[2] Concordia University, Computer Science & Software Dept.,
ormandj@cse.concordia.ca
[3] University of Twente, Information Systems Group
m.daneva@utwente.nl

Abstract. The project effort calculation with a functional size measurement method such as COSMIC can only be properly performed after the "Requirements Analysis" phase in a Project Life Cycle. The goal of this research is to investigate an early and project-level tuned prediction of the product size with the intent to reduce the effect of the 'cone of uncertainty' phenomenon. The lack of size measurement methods which take into account the effect of the product non-functional requirements (NFR) on size also contributes to the above phenomenon. We propose to use the Project Size Unit (PSU) technique for predicting the product (FUR and NFR) size measured in COSMIC functional size units. Such early prediction will lower the cost of size counting the project and minimize the estimation error in the requirements phase. Furthermore, the PSU calculation procedure can be automated, which would further reduce the cost of size counting. The expected advantage of jointly using PSU and COSMIC is the ability to get early estimates of the whole project effort.

Keywords: Project Size, Prediction, COSMIC, Project Size Unit (PSU), Functional User Requirements (FUR), Non-Functional Requirements (NFR).

1 Introduction

Increasingly, business demands require anticipated size estimates, in order to define the needed effort and the related cost (and expected revenues) for a project. However, when dealing with a FSM method, the product functional size can be simply estimated at early stages and not fully counted, as also discussed in the IFPUG CPM [19]. Therefore, the size calculation with a FSM method such as COSMIC can only be properly conducted at the end of the "Requirements Analysis" phase in a Project Life Cycle, having at your disposal an "advanced" *information detail* about the implementation for the software to be developed. Even if there are guidelines about the usage at early stages of FSM methods, the more refined the FUR, the higher the number of functional size units (fsu) obtained. Again, at early stages several non-functional requirements (NFR) must be accomplished (i.e. architectural and setup

R. Dumke et al. (Eds.): IWSM / MetriKon / Mensura 2008, LNCS 5338, pp. 352–361, 2008.

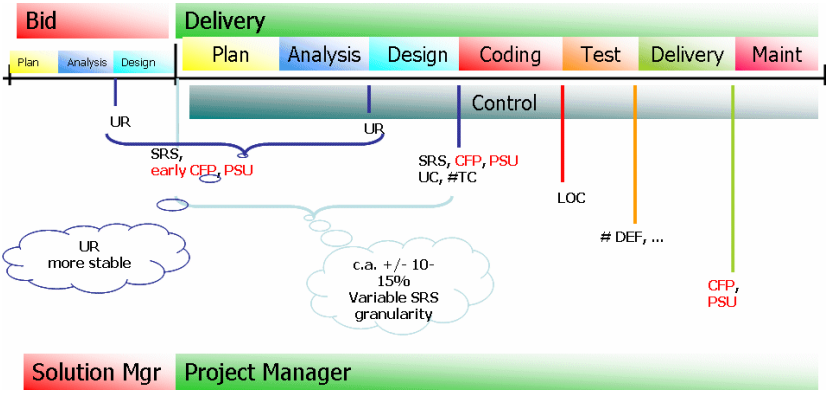


Fig. 1. Sizing measures and possible gathering moments during the SLC

tasks) and would not be considered in the product functional sizing. Since the goal for an estimator is to take care of the whole project boundary, as in Scope Management approaches, a complementary view on the way a project can be sized must be evaluated and introduced (see PMI's PMBOK [22] in the project domain and SouthernScope [23] and NorthernScope [24] in the software engineering domain). Figure 1 summarizes the moments and measures for typically sizing a project during the whole SLC, from the Bid phase on.

The current effort estimation techniques use the *product* functional size of software and not the size of a software *project* as an independent variable [9]. In the early estimation of the overall project effort, however, taking product FURs into account only translated into a (product) functional size, which definitely contributes to a larger MRE (Magnitude of Relative Error) in the early phases; that is, to the '*cone of uncertainty'* phenomenon [21] where the earlier the estimation, the larger the MRE as compared to the final results. The lack of size measurement methods which consider the effects of the product non-functional requirements (NFR) on size also contributes to the above phenomenon; in non-MIS projects. NFRs present a percentage of non-functional effort that can represent up to 50% of the overall project effort.

The aim of this research is to allow for an early and project-tuned prediction of the product size with the intent to reduce the effect of the 'cone of uncertainty'. Therefore, the research question is "How to predict product size for both FUR and NFR from project size before the analysis phase?"

We propose to use the PSU technique for predicting the product (FUR and NFR) size measured in COSMIC [2] [3] functional size units. Such early prediction will lower the cost of size counting the project and minimize the estimation error in the requirements phase. Furthermore, the PSU calculation procedure can be automated, which would further reduce the cost of size counting. Since calculating PSU takes less time, it can easily be used by small and medium enterprises (SMEs) who may not have the time or resources for learning and applying a FSM such as COSMIC. The expected advantage of jointly using PSU and COSMIC is the ability to get early estimates of the whole project effort with predicted CFP in the feasibility study phase. We also expect this would result in a better effort estimation approximation.

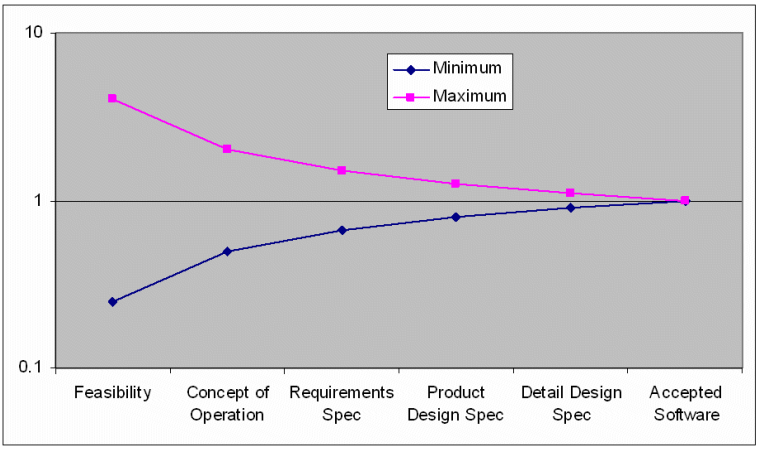


Fig. 2. The cone of uncertainty [21]

In what follows, we first provide background on PSU and COSMIC (see sections 2 and 3). The prediction formula and justification are presented in section 4. The approach is illustrated on student projects' data in section 5. The conclusions and the future research are outlined in section 6.

2 Overview of PSU (Project Size Unit)

PSU was first launched in 2003 as part of a Sw-CMM [16] level 3 certification process in an 80+staff-member organizational unit of a large multinational ICT company. One of the first challenges solved by means of the PSU was to accomplish those requirements from the Software Project Planning key process area which request the estimate efforts and costs (PP, Ac10), taking the overall project scope into account (PP, Ac2)[1]. Since the size calculated with a FSM method should directly refer solely to functional effort (and not to the overall project effort that also includes implicit and explicit project-level requirements, as well as product-level FUR and NFRs), a different process to size a project was put in action. The key idea was to move from the project boundary of the planned/executed activities to a scope extension that includes both FURs (Functional User Requirements) and NFRs.

From a Project Management viewpoint it means considering the whole sum of activities included in a WBS, trying to estimate the total amount of effort from requirements in an early stage. In fact, referring to ISO 9000 [15], the "quality" definition includes both explicit and implicit requirements, where activities and ensuing effort are generated by both and are therefore estimated and planned within the project boundary.

As Figure 3 shows, the goal of the PSU design was to define a new measure at the *project* level for approximating overall "project size" in the early stages.

[1] This also occurs with the newer CMMI-DEV v1.2 [14] model, where the old SPP key process area was simply renamed Project Planning (PP).

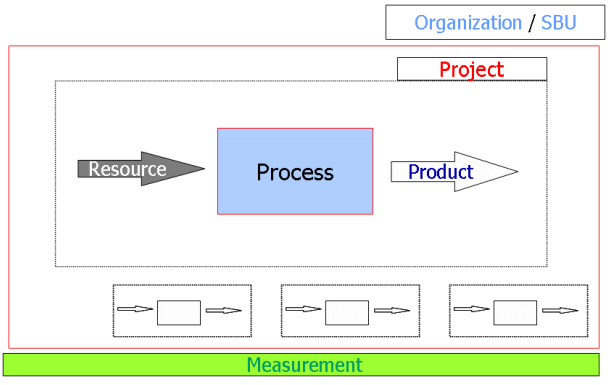


Fig. 3. STAR Taxonomy: measurable entities [13]

Fig. 4. Container (*project*) and content (*product*) [9]

"Project Size" is a term not yet defined in the ISO/IEEE/PMI glossaries. A proposal, according to the above premise, is to define it as "*the size of a software project, derived by quantifying the (implicit/explicit) user requirements referable to the scope of the project itself*" [9]. This term (and our definition) was proposed for inclusion in the next revision of the ISBSG Glossary of Terms [15].

Another example from the real world is the one proposed in Figure 4. Looking at a glass filled with wine, the size of the content (in this case the amount of wine) is not the size of its container (the glass). Applying this image to the entities represented in Figure 3, how could the product's (variable) content size allow for estimation of the size of its container (project)?

Unlike a FSM method, PSU needs an experiential/analogous estimate to produce a more refined estimate, compared with the 'organizational memory' (Project Historical Database - PHD). The PSU-counting is based on the WBS project tasks by three types: management (M), quality (Q) and technical (T) tasks. The T-tasks refer to the primary processes, while the M/Q-tasks refer to the organizational and support processes.

Each task is characterized by its complexity, which is measured by the effort that task requires. The greater the effort required for a task, without any control/milestone in the middle, the more complex and consequently, the riskier it is, with higher probability to request a re-plan during the project lifetime. So, the tactic during the drafting for a WBS is to refine it at the right level trying to minimize high-complexity

tasks as much as possible, balancing the distribution of the forecasted effort against the several possible views (by SLC phase; by effort type; by task type, etc.). The PSU formula can be summarized as follows:

$$PSU = \sum_{i=M,Q,T} \sum_{j=H,M,L} task_i * weight_j \quad (1)$$

where the weights' ranges can vary according to the organizational style and definition for creating projects' WBS and can be easily derived by regularly applying the Pareto Analysis on the Project Historical Database (PHD). For detailed procedures, we refer interested readers to the PSU Measurement Manual [3]. By taking care of (at least) two main groups of requirements (FURs and NFRs), it is also possible to derive the final number of PSU as the sum of the PSU_f (calculated from the tasks derived by FURs) and PSU_{nf} (calculated from the tasks derived by NFRs).

A recent case study using 33 projects that were also sized with IFPUG FPA v4.2 and COSMIC-FFP v2.2 [25] showed a good PSU prediction capability using a standard weighting system. The periodical update of the weighting system results in obtaining a better fit for newer estimates, moving away from the way estimators within the organization previously obtained results and further reducing episodes of the 'cone of uncertainty' as described above.

Again, since the input for calculating PSU are the tasks composing the project WBS, it is possible, as opposed to the FSM method, to easily automate its calculation under any project management software tool [18][2], even on the intensive human-based activity of elicitation and refinement of FUR. Plenty of project data and attributes stored within the software project management tool can be managed with an export utility in XML/CVS format in order to facilitate the creation and maintenance of the organizational PHD, moving progressively from experience/analogy-based estimates towards regression analysis-based ones.

3 Overview of COSMIC

The COSMIC measurement method conforms to all ISO requirements (ISO 14143-1 [20]) for functional size measurement. COSMIC focuses on the "user view" of functional requirements, and is applicable throughout the development life cycle, from the requirements analysis phase right through to the implementation and maintenance phases.

The process of measuring software functional size using the COSMIC method implies that the software functional processes and their triggering events be identified. These are available in the analysis phase. In COSMIC, the unit of measurement is the data movement, which is a base functional component that moves one or more data attributes belonging to a single data group. It is denoted by the symbol *CFP* (Cosmic Function Point). Data movements can be of four types: Entry, Exit, Read or Write. The functional process is an elementary component of a set of user requirements triggered by one or more triggering events, either directly or indirectly, via an actor.

[2] A first implementation under an Open Source Software (OSS) was done with GanttProject (www.ganttproject.org) v2.0.3 [26].

In [5] COSMIC is used for sizing NFR stated in verifiable terms. This means that NFR are stated in terms of crisp indicators with defined acceptable values; thus, it is possible to verify the satisfaction level of those NFR by comparing the acceptable values with the actual achieved values.

4 Predicting CFP with PSU

Prediction determines the likely future values of product measures based on existing measures of the same product. For the purpose of early size prediction we need to define a relationship between product size CFP and project size PSU by requirements type, that is, FUR and NFR. Such relations will allow for: (1) reducing the size measurement effort at this early stage; (2) allowing for accurate size prediction of all NFR, including those which are not (yet) stated in measurable terms.

As stated in section 2, PSU respects the additive property, thus the following equation is valid theoretically from the representational theory of measurement point of view: $PSU=PSU_f + PSU_{nf}$ hence the scale type of the PSU is at the least interval.

On the other side, the addition of the CFP size values ($CFP=CFP_f+ CFP_{nf}$) is also theoretically valid because COSMIC size has a unique unit of measurement, the CFP. Thus the COSMIC size measure is at least on the ratio scale. Consequently, the admissible transformation between the size units CFP and PSU is of type **M'=k*M+b (k>0)** [10]**,** which justify the following relations:

$$PSU_{f} = DataMovementPSU_{fSize} * CFP\_f + b1 \quad (2)$$

$$PSU_{nf} = DataMovementPSU_{nfSize} * CFP\_nf + b2 \quad (3)$$

For further discussion on the scale types and the representational theory of measurement, see [10].

The CFP_f and CFP_{nf} can be predicted in the planning phase of the new project from the actual values of PSU_f and PSU_{nf} and the DataMovementPSUfSize and $DataMovementPSU_{nfSize}$ derived using regression analysis of the PSU and CFP data, where $DataMovementPSU_{fSize}$ and $DataMovementPSU_{nfSize}$ serve as adjustment factors related to the project size.

The $DataMovementPSU_{fSize}$, $DataMovementPSU_{nfSize}$, b1 and b2 can also be derived from the PHD data on CFP_f , CFP_{nf} , PSU_f and PSU_{nf} by using Monte Carlo simulation [11]. Monte Carlo simulation is a problem-solving technique used to approximate the probability of certain outcomes by running multiple trial runs, called simulations; using random variables. We chose it for this research, because JPL and THAAD [27] recommended its use as a solution to deal with uncertainty in software project estimation. For example, these researchers and the last author [12] have deployed it to approach the inherent uncertainty of cost factors. The key advantage of Monte Carlo simulation is that - by collecting samples of the output variables for each run, it helps the estimation analysts produce an overall picture of the combined effect of different input variables' distribution on the output of a model.

To deploy the Monte Carlo simulation in our solution proposal, we first have to ascribe a particular distribution type to the input variables in the model (that is, to CFP_f, CFP_{nf}, PSU_f and PSU_{nf}). When we run the model, the distribution attached to

each input variable will be randomly sampled and the result entered into the model. Repeatedly running the model many times (for example 10 000 times) and collecting samples of the output variables for each run will produce an overall picture of the combined effect of different input variables' distribution on the output of the model. The results of the simulation are in the form of a histogram showing the likelihood of obtaining certain output values for the set of input variables and attached distribution definitions.

While our solution proposal is common sense and sounds intuitive, our results of its use are theoretical and require empirical validation. The proposed method for predicting the product (FUR and NFR) size measured in COSMIC [2] [3] functional size units from the PSU data is illustrated on five student web application development projects.

5 Illustration

The approach described in this paper is illustrated on project data collected on a one-term software project given to five teams formed by third-year undergraduate students in the software engineering program at Concordia University. Each team was given the same problem statement describing an online exam management system that can be used by instructors, students, coordinators, markers, and administrators. Among other services, this software allows i) instructors to manage the question pool, the grades, and conduct exams, ii) students to write real and practice exams, view marks, and register for an exam, iii) markers to grade specific sections of an exam, and iv) administrators to manage courses and user accounts.

In the initial planning activity step the students were asked to estimate the effort for each task entry in their WBS charts and later record the corresponding actual effort. The above data collected by the students served as an input to the PSU_f and PSU_{nf} calculation process, where each task was classified as M/Q/T and assigned the corresponding complexity based on the task's effort estimation and risk assessment and using 4-level complexity schema (H/MH/ML/L).

PHD data collection. The initial start-up hypothesis of the PSU calculation process assumed the weights of the tasks by experience, before evaluating the projects' WBS structure. The low correlation with PSU_f showed that such weighting assignment is unreliable for planning purposes. The weights were recalculated to incorporate the teams' WBS structures and the complexity distribution under the representational constraint weight(L)<weight(ML)<weight(MH)<weight(H). The weights adjustment resulted in a strong statistical relationship (R^2=0.69) between the PSU_f, PSU_{nf} and the tasks evaluation which proved the formula adequate for planning purposes. Additional analysis was carried out with a feasible subdivision into 5 effort ranges for classifying complexity, instead of the previously considered 4-level weighting system (H/MH/ML/L), but verification of the estimation validity of such a model was slightly lower than using 4 levels (with the same data). As a result, it was concluded that i) the 4-level weighting system is statistically more appropriate for planning purposes in this project, and ii) such a system and the PSU data could be used for comparison with

Table 1. Summary of the Student Project Historical Data

Groups	PSU_{total}	PSU_{nf}	PSU_f	CFP_{total}	CFP_{nf}	CFP_f
A	102	60	42	68	15	53
B	77	43	34	131	32	99
C	28	21	7	114	29	85
D	40	25	15	184	22	162
E	97	56	41	147	32	115

COSMIC data as explained in the rest of this section. The size of the FURs developed by each team was measured using the COSMIC method as CFP_f. The size CFP_{nf} of NFR where calculated as described in [6]. Table 1 presents the summary of the PSU and CFP calculations.

Regression analysis results. Formulas (4) and (5) calculated on the project historical data for the 5 projects listed in Table 1 describe the statistical dependencies between the pairs CFP_f, PSU_f and CFP_{nf} , PSU_{nf} :

$$PSU_f = -0.148 * CFP_f + 43 \tag{4}$$

$$PSU_{nf} = -0.47 * CFP_{nf} + 53.28 \tag{5}$$

6 Conclusions and Prospects

This paper aims to resolve one of the major issues, namely the challenge of predicting CFP of FUR and (more importantly) NFR moving from project scope knowledge captured in PSU estimates. A significant outcome is that the FUR and NFR functional size can be predicted from the PSU earlier in software planning, which will help managers in realistically scheduling project milestones. Moreover, the productivity analysis can be performed precisely from the predicted CFP size and the estimated effort. Other advantages of using PSU in this research are:

- *general-purpose*; PSU can be used on all kinds of projects (i.e. service, building, performing arts...).
- can be *automated* under various project management tools, also integrating other useful project information for an XML export easily creating the organizational PHD [18], since the measurable entities are tasks.

In future, we plan to investigate its applicability in real-life project settings. We are also aware of related validity concerns [10] and plan a series of case studies to test our approach, to properly evaluate its validity and to come up with an improved version of it.

References

[1] Albrecht, A.J.: Measuring Application Development Productivity. In: Proc. Joint SHARE/GUIDE/IBM Application Development Symposium, pp. 83–92 (1979)

[2] IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications, Software Engineering Standards Committee of the IEEE Computer Society (1998)

[3] Abran, A., Desharnais, J.-M., Oligny, S., St-Pierre, D., Symons, C.: COSMIC FFP – Measurement Manual (COSMIC implementation guide to ISO/IEC 19761:2003). École de technologie supérieure – Université du Québec, Montréal (2003), http://www.cosmicon.com

[4] ISO/IEC 19761. Software Engineering – COSMIC-FFP – A Functional Size Measurement Method. International Organization for Standardization – ISO, Geneva (2003)

[5] Albrecht, A.J., Gaffney, J.E.: Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. IEEE Trans. Software Eng. SE-9(6), 639–648 (1983), http://www.bfpug.com.br/Artigos/Albrecht/Albrecht_Gaffney.pdf

[6] Kassab, M., Ormandjieva, O., Daneva, M., Abran, A.: Non-Functional Requirements: Size Measurement and Testing with COSMIC-FFP. In: Cuadrado-Gallego, J.J., Braungarten, R., Dumke, R.R., Abran, A. (eds.) IWSM-Mensura 2007. LNCS, vol. 4895, pp. 168–182. Springer, Heidelberg (2008)

[7] Kassab, M., Ormandjieva, O., Daneva, M., Abran, A.: Towards a Scope Management of Non-Functional Requirements in Requirements Engineering. In: Proceedings of MeReP: Workshop on Measuring Requirements for Project and Product Success, Palma de Majorca, Spain, November 6, 2007, pp. 88–99. University of Heidelberg Press (2007), http://www-swe.informatik.uni-heidelberg.de/home/events/MeRePDocs/

[8] Buglione, L.: Project Size Unit (PSU) - Measurement Manual, version 1.21 (November 2007), http://www.geocities.com/lbu_measure/psu/psu-mm-121e.pdf

[9] Buglione, L.: Some Thoughts on Productivity in ICT projects, WP-2008-01, White Paper, version 1.2 (March 2008), http://www.geocities.com/lbu_measure/fpa/fsm-prod-120e.pdf

[10] Buglione, L.: Improving Estimation by Effort Type Proportions. Software Measurement News 13(1), 55–64 (2008), http://ivs.cs.uni-magdeburg.de/sw-eng/us/giak/SMN-08-1.htm

[11] Fenton, N., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach, 2nd edn. International Thompson Computer Press (1997) ISBN 0-534-95425-1

[12] Daneva, M.: Approaching the ERP Project Cost Estimation Problem: an Experiment. In: 1st International Symposium on Empirical Software Engineering and Measurement, p. 500. IEEE Press, New York (2007)

[13] Buglione, L., Abran, A.: ICEBERG: A Different Look at Software Project Management, IWSM 2002 in Software Measurement and Estimation. In: Proceedings of the 12th International Workshop on Software Measurement (IWSM 2002), Magdeburg, Germany, October 7-9, 2002, pp. 153–167. Shaker Verlag (2008) (2008-05-23), http://www.lrgl.uqam.ca/publications/pdf/757.pdf

[14] CMMI Product Team, CMMI for Development, Version 1.2, CMMI-DEV v1.2, CMU/SEI-2006-TR-008, Technical Report, Software Engineering Institute (August 2006) (2008-05-23), http://www.sei.cmu.edu/publications/documents/06.reports/06tr008.html

[15] IEEE, Software & Systems Engineering Vocabulary (SEVOCAB), IEEE Computer Society & ISO/IEC JTC1/SC7 (2008-05-23), http://pascal.computer.org/sev_display/index.action

[16] Paulk, M.C., Weber, C.V., Garcia, S.M., Chrissis, M.B., Bush, M.: Key Practices of the Capability Maturity Model Version 1.1, Software Engineering Institute, CMU/SEI-93-TR-025 (February 1993) (2008-05-23), http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr25.93.pdf

[17] ISBSG, Glossary of Terms, version 5.9.1, International Software Benchmarking Standards Group (28/06/2006) (2008-05-23), http://www.isbsg.org/html/Glossary_of_Terms.doc

[18] Buglione, L.: Project Size Unit (PSU) – Calculation feature in Project Management tools - Requirements, v1.0, PSU-AU-1.00e (December 2006) (2008-05-23), http://www.geocities.com/lbu_measure/psu/psu.htm

[19] IFPUG, Function Points Counting Practices Manual (release 4.2), International Function Point User Group (January 2004) (2008-05-23) , http://www.ifpug.org

[20] ISO/IEC, International Standard 14143-1 - Information Technology - Software Measurement - Functional Size Measurement - Part 1: Definition of Concepts (February 2007)

[21] Boehm, B.: Software Engineering Economics. Prentice-Hall Inc., Englewood Cliffs (1981)

[22] Project Management Institute, A Guide to the Project Management Body of Knowledge, 3rd edn. (2004), ANSI/PMI 99-001-2004, ISBN 1-930699-45-X

[23] Victoria Government, SouthernScope (2007) (2008-05-23), http://www.egov.vic.gov.au/index.php?env=-innews/detail:m1816-1-1-8-s-0:n-832-1-0-

[24] FISMA, NorthernScope (2007) (2008-05-23), http://www.fisma.fi/in-english/scope-management

[25] Buglione, L., Cuadrado-Gallego, J.J., Gutiérrez de Mesa, J.A.: Project Sizing and Estimating: A Case Study using PSU, IFPUG and COSMIC. In: Proceedings of IWSM/Metrikon/Mensura 2008, Munich, Germany, November 18-19, 2008. Springer, Heidelberg (2008)

[26] Biagiotti, C.: Migliorare gli aspetti di stima e pianificazione di un progetto attraverso la customizzazione di un tool OpenSource di Project Management, University of Perugia, Tesi di Laurea, Perugia, Italy (July 2007)

[27] McDonald, P., Giles, S., Strickland, D.: Extensions of Auto-Generated Code and NOSTROMO Methodologies. In: Proc. of 19th Int. Forum on COCOMO, Los Angeles, CA (2001)

Author Index