

*Commenced Publication in 1973*

Founding and Former Series Editors:

Gerhard Goos, Juris Hartmanis, and Jan van Leeuwen

## Editorial Board

David Hutchison

*Lancaster University, UK*

Takeo Kanade

*Carnegie Mellon University, Pittsburgh, PA, USA*

Josef Kittler

*University of Surrey, Guildford, UK*

Jon M. Kleinberg

*Cornell University, Ithaca, NY, USA*

Alfred Kobsa

*University of California, Irvine, CA, USA*

Friedemann Mattern

*ETH Zurich, Switzerland*

John C. Mitchell

*Stanford University, CA, USA*

Moni Naor

*Weizmann Institute of Science, Rehovot, Israel*

Oscar Nierstrasz

*University of Bern, Switzerland*

C. Pandu Rangan

*Indian Institute of Technology, Madras, India*

Bernhard Steffen

*University of Dortmund, Germany*

Madhu Sudan

*Massachusetts Institute of Technology, MA, USA*

Demetri Terzopoulos

*University of California, Los Angeles, CA, USA*

Doug Tygar

*University of California, Berkeley, CA, USA*

Gerhard Weikum

*Max-Planck Institute of Computer Science, Saarbruecken, Germany*

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

# Software Process and Product Measurement

International Conference, IWSM-Mensura 2007  
Palma de Mallorca, Spain, November 5-8, 2007  
Revised Papers

## Volume Editors

Juan J. Cuadrado-Gallego  
University of Alcalá  
Department of Computer Science  
Edificio Politécnico, Autovía A2, Km. 31,7, 28805 Alcalá de Henares, Spain  
E-mail: jjcg@uah.es

René Braungarten  
Reiner R. Dumke  
Otto-von-Guericke-Universität Magdeburg  
Fakultät für Informatik  
Institut für Verteilte Systeme  
AG Softwaretechnik  
Universitätsplatz 2, 39106 Magdeburg, Germany  
E-mail: {braungar, dumke}@ivs.cs.uni-magdeburg.de

Alain Abran  
Université du Québec  
École de technologie supérieure  
Département de génie logiciel et des TI  
1100, rue Notre-Dame Ouest, Montréal, Québec H3C 1K3, Canada  
E-mail: Alain.Abran@etsmtl.ca

Library of Congress Control Number: 2008933374

CR Subject Classification (1998): D.2.8, D.2

LNCS Sublibrary: SL 2 – Programming and Software Engineering

ISSN 0302-9743  
ISBN-10 3-540-85552-1 Springer Berlin Heidelberg New York  
ISBN-13 978-3-540-85552-1 Springer Berlin Heidelberg New York

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

Springer is a part of Springer Science+Business Media  
springer.com

© Springer-Verlag Berlin Heidelberg 2008  
Printed in Germany

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India  
Printed on acid-free paper SPIN: 12320127 06/3180 5 4 3 2 1 0

# Preface

Since 1990 the International Workshop on Software Measurement (IWSM) has been celebrated annually in Montréal (Québec), Canada, and different places all over Germany by turns. The Montréal editions were organized by the Software Engineering Research Laboratory (GELOG)<sup>1</sup> of the École de technologie supérieure (ÉTS) at the University of Québec at Montréal (UQAM), which is directed by Professor Alain Abran. The German editions were organized jointly by the Software Measurement Laboratory (SMLAB)<sup>2</sup> of the Otto-von-Guericke-University Magdeburg, Germany, which is directed by Professor Reiner R. Dumke; and the German-speaking user association for software metrics and effort estimation (DASMA e. V.)<sup>3</sup>. Partially, the editions of IWSM were held jointly with the DASMA Software Metrik Kongress (MetriKon).

Organized by an initiative of José Javier Dolado<sup>4</sup> from the University of the Basque Country at San Sebastian and Juan J. Cuadrado-Gallego<sup>5</sup> from the University of Alcalá in Madrid the first edition of the International Conference on Software Measurement (*Mensura*) could be convened in Cádiz, Spain in 2006. Motivated by this success and with the first edition of *Mensura* finding special approval, the organizers of IWSM and *Mensura* decided to complement each other and, thus, to organize the next conference edition together. In November 2007, the typical convention month for both conferences, that joint conference was held in Palma de Mallorca, Spain.

This volume is the post-proceedings of the IWSM-Mensura 2007 conference and consists of a set of 16 final papers selected from the conference. Each one of these papers has been thoroughly revised and extended in order to be accepted for this edition. The IWSM-Mensura Steering Committee is very proud to have obtained the approval of Springer to publish the first edition of the joint conference post-proceedings in the prestigious *Lecture Notes in Computer Sciences* (LNCS) series and hope to maintain this collaboration for the future editions of the conference.

February 2007

Juan J. Cuadrado-Gallego

---

<sup>1</sup> <http://www.lrgl.uqam.ca/>

<sup>2</sup> <http://ivs.cs.uni-magdeburg.de/sw-eng/us/>

<sup>3</sup> <http://www.dasma.org/>

<sup>4</sup> <http://www.sc.ehu.es/dolado/>

<sup>5</sup> <http://www.cc.uah.es/jjcg/>

# Organization

## General Chairs

Reiner R. Dumke	Otto-von-Guericke-University, Magdeburg, Germany
José Miguel Toro	University of Seville, Spain

## Program Committee Chairs

Alain Abran	University of Québec, Montréal, Canada
Antonia Mas	University of the Balearic Islands, Palma de Mallorca, Spain

## Program Committee

Moteoi Azuma	University of Waseda, Japan
Manfred Bundschuh	DASMA e.V., Germany
David Card	DNV IT Global Services, USA
François Coallier	ÉTS, Montréal, Canada
Juan J. Cuadrado-Gallego	University of Alcalá, Spain
Jean-Marc Desharnais	ÉTS, Montréal, Canada
Ton Dekkers	Shell Information Technology, The Netherlands
José Javier Dolado	University of the Basque Country, Spain
Christof Ebert	vector Consulting, Stuttgart, Germany
Khaled El-Emam	University of Ottawa, Ottawa, Canada
Felix García	University of Castilla-La Mancha, Spain
Elena García-Barriocanal	University of Alcalá, Spain
George Ghinea	Brunel University, UK
Naji Habra	FUNDP, Namur, Belgium
Nadine Hanebutte	University of Idaho, USA
Mark Harman	Kings College, London, UK
Rachel Harrison	Stratton Edge Consulting Ltd., UK
Ali Idri	INSIAS, Morocco
Natalia Juristo	Politechnic University of Madrid, Spain
Adel Khelifi	ALHOSN University, Abu Dhabi, UAE
Pedro Lara	University Europea de Madrid, Spain
Miguel Lopez	University of Namur, Belgium
Mathias Lother	Robert Bosch GmbH, Germany
Hakim Lounis	University of Québec, Montréal, Canada

Fernando Machado	Catholic University of Uruguay, Montevideo, Uruguay
John McGarry	U.S. Army, USA
Roberto Meli	DPO, Italy
Pam Morris	Total Metrics, Australia
John C. Munson	University of Idaho, USA
Olga Ormandjieva	Concordia University, Montréal, Canada
Oscar Pastor	Technical University of Valence, Spain
Mario Piattini	University of Castilla-La Mancha, Spain
Tony Rollo	Software Measurement Services Ltd., UK
Salvador Sánchez	University of Alcalá, Spain
Daniel Rodríguez	University of Alcalá, Spain
Miguel-Ángel Sicilia	University of Alcalá, Spain
Maya Daneva	University of Twente, The Netherlands
Manoranjan Satpathy	General Motors, India
Andreas Schmietendorf	Berlin School of Economics, Germany
Manuel Serrano	University of Castilla-La Mancha, Spain
Marcio Silveira	EDS, Brazil
Harry Sneed	SES, Munich/Budapest, Hungary
Esperança Amengual	Univ. of the Balearic Islands, Palma de Mallorca, Spain
Stanimir Stojanov	University of Plovdiv, Bulgaria
Hannu Toivonen	Nokia, Finland
Claes Wohlin	Blekinge Institute of Technology, Sweden
David Zubrow	Software Engineering Institute, USA
Horst Zuse	Technical University Berlin, Germany
Marcus Ciolkowski	Fraunhofer IESE, Kaiserslautern, Germany
Javier Aroba	University of Huelva, Spain
Dietmar Pfahl	University of Calgary, Canada
Andreas Jedlitschka	Fraunhofer IESE, Kaiserslautern, Germany

## Special Sessions/Workshop Chairs

Mercedes Ruiz	University of Cádiz, Spain
Luis Fernández	University Europea de Madrid, Spain
Luigi Buglione	Engineering.it S.p.A., Italy
Luca Santillo	Independent consultant, Italy
Sylvie Trudel	CRIM, Canada

# Table of Contents

Improving the Quality of Information for Software Project Management .....	1
<i>Michael Berry and Chris S. Johnson</i>	
Software Cost Estimation Models Using Radial Basis Function Neural Networks .....	21
<i>Ali Idri, Azeddine Zahi, Emilia Mendes, and Abdelali Zakrani</i>	
Towards an Early Usability Evaluation for Web Applications .....	32
<i>Jose Ignacio Panach, Nelly Condori-Fernández, Francisco Valverde, Nathalie Aquino, and Oscar Pastor</i>	
An Empirical Study of Process and Product Metrics Based on In-process Measurements of a Standardized Requirements Definition Phase .....	46
<i>Yoshiki Mitani, Tomoko Matsumura, Mike Barker, Seishiro Tsuruho, Katsuro Inoue, and Ken-Ichi Matsumoto</i>	
Preliminary Results in a Multi-site Empirical Study on Cross-Organizational ERP Size and Effort Estimation .....	60
<i>Maya Daneva</i>	
Do Base Functional Component Types Affect the Relationship between Software Functional Size and Effort? .....	72
<i>Cigdem Gencel and Luigi Buglione</i>	
Experiences on Using Software Experiments in the Validation of Industrial Research Questions .....	86
<i>Dominik Gessenharter, Alexander-Marc Merten, Alexander Raschke, and Nicolas Fernando Porta</i>	
How to Measure Agile Software Development .....	95
<i>Martin Kunz, Reiner R. Dumke, and Andreas Schmietendorf</i>	
Assessment of Software Process and Metrics to Support Quantitative Understanding .....	102
<i>Ayça Tarhan and Onur Demirors</i>	
Developing and Applying a Consolidated Evaluation Framework to Analyze Test Process Improvement Approaches .....	114
<i>Ayaz Farooq and Reiner R. Dumke</i>	
Using Controlled Experiments for Validating UML Statechart Diagrams Measures .....	129
<i>José A. Cruz-Lemus, Marcela Genero, and Mario Piattini</i>	

Adapting the COSMIC Method for Evaluating the Functional Size in PRiM .....	139
<i>Gemma Grau</i>	
Implementing Software Measurement Programs in Non Mature Small Settings .....	154
<i>María Díaz-Ley, Félix García, and Mario Piattini</i>	
Non-Functional Requirements Size Measurement Method (NFSM) with COSMIC-FFP .....	168
<i>Mohamad Kassab, Olga Ormandjieva, Maya Daneva, and Alain Abran</i>	
Assessment of Real-Time Software Specifications Quality Using COSMIC-FFP .....	183
<i>Manar Abu Talib, Adel Khelifi, Alain Abran, and Olga Ormandjieva</i>	
Analysis of Software Functional Size Databases .....	195
<i>Juan J. Cuadrado-Gallego, Miguel Garre, Ricardo J. Rejas, and Miguel-Ángel Sicilia</i>	
<b>Author Index</b> .....	203



# Improving the Quality of Information for Software Project Management

Michael Berry<sup>1</sup> and Chris S. Johnson

University of Technology, Sydney, Australia  
mberry@it.uts.edu.au, chrisj@it.uts.edu.au

**Abstract.** *Background.* The goal of software measurement and analysis is to produce information for the people performing technical and managerial tasks. Information-centric assessment can identify opportunities to better meet that goal. This paper presents a study into the assessment of Information Quality for Software Project Management using generic and targeted information quality models. The setting for the study is a university software project management subject in which the students must complete a major exercise as part of the subject that attempts to replicate real-world project planning. *Methods.* After the exercise has been completed, the students are surveyed about the quality of the information with which they were supplied to complete the project planning exercise. The survey instrument is based on a generic model of information quality and a targeted model that focuses on the information for specific planning tasks. The study has been run with two cohorts of students and was run with a third cohort at the end of May 2007. *Results.* Improvement opportunities were identified through the 2006 assessment. These improvements were implemented for the third student cohort and the success of these improvements evaluated. The information quality assessment methods used in this study are likely to be of benefit to software project managers by providing the means to identify and remedy deficiencies in the information that they use to carry out their activities.

## 1 Introduction

This paper reports on part of a longitudinal research program that seeks to address the following research question: *What models and methods are appropriate for the formative assessment of the information support provided to software project managers?* The focus of this paper is on the work conducted in 2006 and the follow-up study in 2007. This work examines the effectiveness of different models for use in information quality assessments for the purpose of improvement. Instruments, based on different models of information quality, are used to assess the same set of information that is representative of the information that might be produced by software measurement and analysis. One model of Information Quality is based on a generic model developed by Lee, Strong, Kahn and Wang [11]. The second Targeted Information Quality model was developed using the Targeted Assessment method described in [3] and is designed for evaluations of the information needed for specific project planning tasks.

---

<sup>1</sup> Michael Berry is principal of EBSE Australia [evbase@ebse.com.au](mailto:evbase@ebse.com.au).

## 1.1 Key Terms

In software engineering, the Measurement and Analysis process is the means of producing *Information Products* for project management, therefore the term, **M&A**, has been used to label the information production process. Assessment of information quality uses an instrument that is based on an *Information Quality Model*. These terms are defined below.

### Information Products

Information products<sup>2</sup> are intended to satisfy an information need arising out a process, activity or task. In this study, the information products are the documents provided to the students – a functional specification and documents containing project constraints and data for use in project estimation.

### Information Quality

*Abbreviation used in this paper: IQ*

Information Quality (**IQ**) is a *Measurable Concept*<sup>3</sup> that describes an abstract relationship between the attributes of information and the needs of the people who use the information. In this study, we measured the IQ of the information products provided to the students.

**IQ** measurement may employ objective and subjective metrics. Objective assessment measures intrinsic properties of the information product, for example, volume of text, number of figures, Flesch Reading Ease score<sup>4</sup>. Subjective assessment measures the perceptions of the stake-holders to the information product. Their perceptions are based on many properties of the information product such as relevance, timeliness, completeness and reliability with respect to their information needs. In this study, subjective measurement is used to evaluate the IQ of the information products given to the students.

### Information Quality Models

*Abbreviation used in this paper: IQ Model*

An *Information Quality Model* structures the IQ measurable concept<sup>5</sup> by defining the relationship between attributes of information products and information needs. Generic models state that, in general, the same relationships always hold; e.g. a generic model might say that, no matter what process is being performed, the relevance, timeliness, completeness and reliability of the information product are important factors in the satisfaction of information need. Targeted models, on the other hand, state that, in a particular context and when performing a particular process, the satisfaction of the information need will depend on a specific set of information product attributes. A targeted model may include weights for each attribute so that their relative importance can be brought into the algorithm that calculates the indicator for identifying the best opportunities for improvement. In this study, a generic model and a targeted model are used to evaluate the IQ of information products given to the students.

---

<sup>2</sup> This term is used in ISO/IEC 15939: Software Measurement Process [8].

<sup>3</sup> This term is taken from ISO/IEC 15939: Software Measurement Process [8].

<sup>4</sup> Flesch, R., A new readability yardstick. *Journal of Applied Psychology*. 32(3) 221-233 1948.

<sup>5</sup> See definition in ISO/IEC 15939 [8].

### Formative Assessment

**Formative Assessment** is assessment carried out with the specific goal of identifying opportunities for improvement. It is contrasted with Summative Assessment where the goal is to assign a score or rating to the object of interest. **Assessment of information quality** uses an instrument that is based on an **Information Quality Model**.

### Information-centric Assessment

Frederiksen, H. D. and Mathiassen, L. [7] use the term **Information-centric Assessment** for assessment that focuses on the information products of M&A. It can be contrasted with Framework-centric Assessment where the focus is on the organisational structure within which M&A is performed and with Process-centric Assessment. **Assessment of information quality** uses an instrument that is based on an **Information Quality Model**.

## 1.2 The Problem Addressed by This Study

The problem being addressed is how to improve the quality of information provided to software engineers and managers through software measurement and analysis. Information-centric assessment has the potential to identify deficiencies in the information provided and thereby lead to improvements. However, there has been the limited uptake of this form of assessment by the software engineering industry. Among reasons for this may be immaturity of the models and methods, complexity of the technology [7], perceptions that it is only appropriate for high capability organisations, cost/benefit issues and slow diffusion. Targeted assessment, in particular, requires an information quality model tailored for the particular process, context and people. First the model has to be built and then the assessment instrument constructed to fit the model. This obviously requires time and resources. Furthermore, many models and many instruments are required to cover the set of software processes performed in an organisation. In contrast, assessment using a generic model of information quality can employ a standard instrument and take advantage of economies of scale. If assessment using a generic model is adequate for improvement purposes, then it may be preferred over targeted assessment, particularly if assessment using a generic model is equally effective.

## 1.3 Scope of the Study

This is a longitudinal empirical study that is a component of an ongoing research program into the effectiveness of models and methods for assessing information products for use in software engineering processes. This component focuses on Software Project Management.

1. In 2003, students in a software project management course evaluated the quality of the information that they had been given in order to carry out a project planning assignment. The analyses produced from the students' evaluations were then evaluated by a panel of academics, project managers and measurement specialists to determine the quality of the analyses for the purpose of improving information quality.

2. In November 2006, a second cohort of students evaluated a similar set of information using the same instrument. Based on their feedback, improvements were made to the information products to be provided to the next student cohort.
3. In May 2007, a third cohort of students evaluated the improved information product. Their evaluation was an indicator of the effectiveness of the improvements made to the information product. It also identified new opportunities for improvement.

#### 1.4 The 2003 Information Quality Study

The respondents in the 2003 study were 230 students who had taken a third year undergraduate course in Software Project Management. As part of their course, the students had completed a practical exercise in project planning in groups of up to five. At the end of the course, they were asked to evaluate the quality of the information that they been given for the exercise. The volunteers received no additional course credit for participating in the evaluation. Eighty students began the IQ evaluation but only thirty-five students completed all questions. There was no way of determining why respondents dropped out or why 150 students did not volunteer to respond. There is therefore a significant threat to the validity of the data that the respondents were not representative of the population.

Analysis of the responses to the Generic IQ Instrument identified the following high priority improvement actions:

- Improve *Interpretability*. There was a strong indication that the students did not really know how to interpret the information that they were given in the context of the tasks that they had to perform.
- Improve *Ease of Use*. There was a strong indication that the students did not really know how to manipulate the data that they were given in the context of the tasks they performed.

Analysis of the responses to the Targeted IQ instrument indicated that improving the quality of the information supporting the tasks of **Sizing** and **Estimation** presented the best opportunities. Supporting the tasks of scheduling, task sequencing and task assignment was of a lower priority.

The analyses from the 2003 study were assessed for their information quality for the purpose of improvement by an expert panel of academics, project managers and measurement specialists, using an instrument based on the AIMQ generic model of IQ[11]. The experts' preferences varied by the information quality dimension being rated. On the four key dimensions for carrying out improvements ("Appropriate Amount", "Completeness", "Ease of Use" and "Interpretability"), there was a distinct preference for the Targeted Analysis. On the IQ dimension of "Understandability", approximately one third had no preference; one third preferred the Generic Analysis and one third preferred the Targeted Analysis. The conclusion was that while Targeted Method had potential, the value of using the analyses from the 2003 study could not be demonstrated because no improvements were actually made that were based on the analyses. To demonstrate this it would be necessary to complete a cycle of assessment, improvement and re-assessment.

## 2 Models of Information Quality

The incremental *Plan-Do-Check-Act* improvement paradigm incorporated in ISO/IEC standard 15939 – Software Measurement Process [8] and the SEI’s CMMI “Measurement and Analysis” process area [16] provide frameworks for conducting meta-measurement. However, these process-oriented approaches to meta-measurement do not provide a complete view of measurement performance in that they pay insufficient attention to the relationship between a human being’s cognitive processes and the information products produced by M&A. Some researchers have examined this relationship: for example, Rivet discussed a cognitive-based approach to designing measurement frameworks [13] and Baddo and colleagues [1] have evaluated client satisfaction with software measurement as part of a wider study into sentiments towards software process improvement.

Information-centric assessment approaches, such as the work of a group of MIS researchers centred on the Information Quality program at MIT [9-11] may provide a wider view of M&A. Lee et al [11] have developed a model of Information Quality (IQ) that contains 15 constructs (termed IQ Dimensions). The model is the basis for the AIMQ instrument which can be used to identify problems with an organisation’s information systems and/or to assess the systems against benchmarks. The instrument was validated using a test group of managers and was shown to have acceptable reliability, as measured by Cronbach’s alpha. The AIMQ instrument is simple to deploy and analysis of the collected data is straight-forward. The instrument, therefore, appears to offer a cheap and easy method to evaluate and benchmark software measurement. However, analyses prepared from the generic AIMQ model may provide insufficient support for improving software measurement. Also, there are suggestions that the empirical method used to derive the IQ dimensions has resulted in a non-orthogonal set of attributes that may contain omissions [6;12]. Definitions of the IQ dimensions that were evaluated in this study are listed in Table 1. AIMQ Model: Definitions of IQ Dimension.

**Table 1.** AIMQ Model: Definitions of IQ Dimension

IQ Dimension	Definition
Appropriate Amount	The extent to which the volume of information is appropriate to the task at hand.
Completeness	The extent to which information is not missing and is of sufficient breadth and depth for the task at hand.
Ease of Use	The extent to which information is easy to manipulate and apply to different tasks.
Interpretability	The extent to which information is in appropriate languages, symbols and units and the definitions are clear.
Relevancy	The extent to which information is applicable and helpful for the task at hand.
Understandability	The extent to which information is easily comprehended.

Targeted assessment of information quality [3] has been specifically developed to support the improvement of software measurement and analysis. Trials of assessments using Targeted IQ model have successfully identified priority areas for improvement, providing clear direction to the “improvers” as to what needs to be improved first and why. However, the Targeted assessment documented in [3] cost two hundred hours of resource time and addressed only one key practice area in the CMM model. This makes it an expensive approach if using a generic model is equally effective.

### 3 The 2006 Information Quality Study

The 2006 IQ study is a replication of the 2003 IQ study [2;4] using respondents studying the same subject at the same university. The project management tasks performed were essentially the same; however the information product that the students were given to complete those tasks differed in that the functional specification concerned a different system. IQ assessments were conducted using generic and targeted models and analyses produced. Improvements were then made to the information provided to the 2007 student cohort. The 2007 cohort was then surveyed to determine whether the improvements had been effective.

#### 3.1 Research Questions

The Research Questions are concerned with the *effectiveness* of the assessment methods; specifically:

RQ1 Does assessment of information products using a Generic IQ model yield information of sufficient quality to support the improvement of those information products? The measure of this capability is whether a set of meaningful recommendations can be based on the data collected with the instrument.

RQ2 Does assessment of information products using a Targeted IQ model yield information of sufficient quality to support improvement of those information products? The measure of this capability is whether a set of meaningful recommendations can be based on the data collected with the instrument.

The quality of the information product was assessed in terms of the support that the information provided for a limited set of project management tasks carried out by students who were simulating “practicing software project managers”. The information was provided in the students’ assignment; in a real software organisation, it would have been a product of the software measurement and analysis process. The students’ assessment provided two analyses of information quality: one generic, one targeted.

#### 3.2 The Respondents

The population being sampled consisted of 162 third year undergraduate students and graduate diploma students who had completed Software Project Management (completion means “took the final exam”). As part of their course, the students had undertaken a practical exercise in project planning. At the end of the course, they were asked to evaluate the quality of the information that they had been given for the exercise. The respondents received additional course credit for participating in the evaluation. The practical exercise had required the students to decompose a functional specification

and then estimate and schedule the software development tasks involved. While students may have been given varying assistance from their tutors on how to carry out the tasks required for the assignment, all students received the same information to be used as input to those tasks. Eighty-six students began the IQ evaluation but only seventy-four students provided usable sets of responses. Partially completed response sets were usable because of the randomized order of presentation of questions. There was no way of determining why respondents dropped out or why some students did not volunteer to respond. The default response was “Don’t know” and was treated as a valid response; however such responses were not used for calculations. The response rate of 53% of students enrolled in the subject was satisfactory and was an improvement on the participation rate in the 2003 study.

**Table 2.** Work Experience

Have you ever been in fulltime work?	UnderGrad Percent	PostGrad Percent
Never worked fulltime	52%	22%
1 - Less than a year	11%	6%
2 - One to two years	13%	17%
3 - More than 2 years and less than 3	11%	0
4 - More than 3 years and less than 4	0	0
5 - More than 4 years	14%	55%
Total	100.0	100.0

The students provided data about their work experience: significantly more Post-graduate respondents had fulltime work experience (Table 2); however, the percentage of these who had **No** managerial experience was the same in both groups at around 70%. Within the total set of students, there were highly significant, strong correlations between whether a student was a post-graduate and their mark on the Assignment (0.372,  $p < .0$ ). Within the set of students who responded, these correlations were even higher (0.506,  $p < .0$ ). Coupled with ANOVA tests that indicate that 18% of the subject results were accounted by whether the student was undergraduate or postgraduate, we concluded that the respondent sample is not homogeneous and that it would be necessary to determine whether Undergraduate and Postgraduate students significantly differed in their IQ responses.

### 3.3 Data Collection

Web forms were used to collect the respondents’ perceptions of the quality of the information they had been given in order to carry out their assigned project management tasks. The use of web forms was a trade-off between the richness of the data that might be collected in interviews and the need to consult widely. Each form produced responses scored on a Likert scale. In addition, each form contained a free-text input

field to capture comments. The demographic form was presented first, and the feedback form, last. The other forms were interspersed and presented in a randomised order to the respondents. The Respondent Demographic test items used a pull-down menu with prescribed categories. The other items used a thirteen-point agree/disagree Likert scale, anchored at each end with the words “Very Strongly Agree” and “Very Strongly Disagree”. The format was that the respondent was presented with an assertion with which they were invited to register the degree of their agreement (or disagreement). A distinct “Don’t know” response was provided as a default value for all test items. The scales for performance and importance were visually aligned in order to suggest to the respondents that they should treat performance and importance as being measured on similar scales.

The choice of a thirteen-point scale needs some explanation: respondent’s perceptions are likely to have continuous distributions where concepts such as performance and importance are concerned. Five-point and seven-point Likert scales can only approximate continuous distributions using an ordinal scale. A thirteen-point Likert scale was used because it could provide a closer approximation to a continuous distribution and would collect data of finer granularity. In an earlier study [3], the seven-point Likert scale had been found to be too gross and the designers of the AIMQ instrument [11] had used an 11 point scale commenting that, *“The use of an 11-point scale is based on previous experience with IQ assessment.”* The use of thirteen points, and not more, was determined by a preferred maximum size for the web form. In fact, if a Java-based slider had been ready in time, it would have been used in preference to the Likert scale..

### 3.4 Generic IQ Test Items

Twenty-five test items evaluated six IQ dimensions taken from the AIMQ generic IQ model [11]; these items are listed in Appendix A. The test items that were used were taken from the AIMQ instrument with minor rephrasing following a pilot study. The response scale was constructed so that low ratings indicated positive sentiments (satisfaction) towards the particular IQ dimension, while high ratings indicated dissatisfaction. Analysis of the responses suggests that apart from the “*Interpretability*” dimension, a reasonably reliable instrument ( $\alpha \geq 0.70$ ) had been produced. It was also noticeable that the test items in the “*Interpretability*” construct had the greatest proportion of “Don’t know” responses and resulted in the smallest number of valid cases out of a possible 86 sets of responses. The reliability measures for the Assessment Instrument for the six AIMQ IQ dimensions that were evaluated are shown in Table 3.

**Table 3.** Generic IQ Instrument Reliability

IQ Dimension	N	N of Items	2006 Reliability (alpha)	2003 Reliability (alpha)
Understandability	74	4	.84	0.83
Completeness	72	3	.84	0.82
Appropriate Amount	74	3	.83	0.82
Ease of Use	68	5	.76	0.72
Interpretability	63	5	.69	0.47
Relevancy	71	4	.90	0.80



### 3.5 Targeted IQ Test Items

Five sets of test items evaluated the information from a targeted IQ perspective, these items are listed in Appendix B. Each set targeted a specific practice that the students would need to carry out to complete their assignment. The targeted practices were identified by reference to the *Specific Practices* in the CMMI model for the “*Project Planning*” process area [16]. The practices that were targeted for characterisation were those for which a significant amount of information is required in order to complete the practice. These were:

1. ***Establish Estimates of Work Product and Task Attributes.***
2. ***Determine Estimates of Effort and Cost.***
3. ***Establish the Budget and Schedule: Develop a schedule and Determine task dependencies***
4. ***Plan for Project Resources.***

For each targeted practice, there were four test items that:

1. Evaluated the respondent’s perception of how well the practice was performed.
2. Measured the importance attached by the respondent to performing the practice well.
3. Evaluated the quality of the information provided in order to carry out the practice.
4. Measured the importance attached by the respondent to having high quality information in order to carry out the practice.

As in [3], each test item was stated in the form of a positive assertion. An additional test item that was based on an AIMQ dimension of IQ was included in each form. The intention was to examine the effectiveness of test items based on a Hybrid IQ model. Because the results are inconclusive, discussion of the Hybrid model has been omitted from this paper. The box below shows the contents of a typical Targeted IQ Form.

The information referred to below is the information that you were provided with in order to <b><i>Establish Estimates of Work Product and Task Attributes</i></b> (CMMI Project Management SP 1.2-1).		
	<b><u>Practice View</u></b>	<b><u>Information View</u></b>
<b>Performance</b>	Sizing of work products was accurately performed.	The methods for estimating the size of work products were effective.
<b>Importance</b>	It is important that work products are accurately sized.	It is important that there are effective methods for estimating the size of work products.

### 3.6 Analysis of Information Quality Using the Generic Model

Whether a respondent was a postgraduate or undergraduate student was not significant. Therefore the two sets of data were aggregated. The ratings for each IQ construct were treated as interval data and means were calculated to characterise the respondents’ levels

of satisfaction along each of the IQ dimensions. See Fig. 1 for a summary of the responses. In the 2003 data there was an inexplicable range of 12 points in the respondents' ratings for each IQ Dimension since all respondents were supposedly assessing the same set of information. However the range is smaller in the 2006 study. Except for the “*Completeness*” construct which is skewed to the left, the frequency distributions are approximately normal. The standard deviation for each construct was approximately 2 rating points.

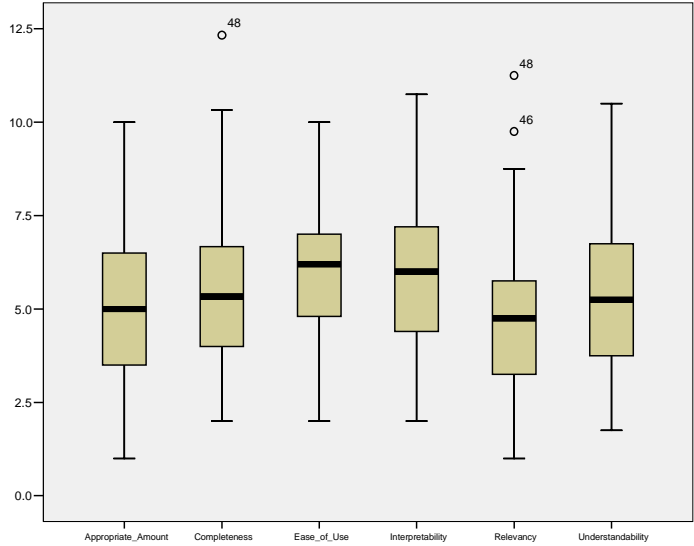


Fig. 1. 2006 IQ Study: Summary by Construct

Table 4 summarises the responses received, by IQ dimension. Scores at the low end of the 1-13 scale suggest satisfaction with the IQ dimension. Compared to the 2003 IQ Scores, the respondents in the 2006 study seemed more satisfied with the quality of the information provided. However, it is noticeable that the ranking of the IQ dimensions in terms of .satisfaction were similar, with the greatest dissatisfaction being recorded for the “*Ease of Use*” and “*Interpretability*” dimensions.

Table 4. Summary of Satisfaction by IQ Dimension

IQ Dimension	Mean (2006)	Std. Dev (2006)	Mean (2003)
<i>Ease of Use</i>	6.0	1.88	7.6
<i>Interpretability</i>	5.9	1.95	7.6
Completeness	5.4	2.2	7.3
Understandability	5.5	2.15	7.2
Appropriate Amount	5.1	2.17	7.0
Relevancy	4.7	2.06	6.4

### 3.7 Analysis of Information Quality Using the Targeted Model

This section provides a summary of the analysis that was prepared from data collected using the targeted model of information quality. The ratings provided by the respondents in response to test items based on the targeted assessment model were treated as interval data and means were calculated to characterise each of the targeted practices in terms of performance of each practice ( $\text{MeanR}^P$ ), the importance of the practice ( $\text{MeanR}^{PI}$ ), the quality of the measurement support ( $\text{MeanR}^M$ ) and the importance of measurement support to perform the practice ( $\text{MeanR}^{MI}$ ). **Higher**  $\text{MeanR}^P$  and  $\text{MeanR}^M$  values **suggest dissatisfaction** with the performance of a practice. **Lower**  $\text{MeanR}^{PI}$  and  $\text{MeanR}^{MI}$  values indicate **more importance** assigned to the practice.

**Table 5.** Performance/Importance by Targeted Practice

	Practice		Information	
	Performance	Importance	Quality	Importance
<b>Targeted Practice</b>	<b><math>\text{MeanR}^P</math></b>	<b><math>\text{MeanR}^{PI}</math></b>	<b><math>\text{MeanR}^M</math></b>	<b><math>\text{MeanR}^{MI}</math></b>
Sizing	5.5	3.2	5	3.4
Estimation	6.1	3.6	6.6	3.9
Scheduling	5.6	3.2	6.8	4.1
Task sequencing	5.1	3.3	5.9	4.5
Task assignment	5.6	3.8	5.2	3.6

An indicator ( $IOpp$ ) was calculated in order to objectively rank the opportunities to improve. The **IOpp** is an indicator of the difference between performance and importance that also takes into account the divergence of opinion between the respondents. The formula is shown below:

$$IOpp = (\text{mean(Performance)} - (\text{mean(Importance)}) / \text{SQRT}((\text{stddev(Performance)} * \text{stddev(Performance)}) + (\text{stddev(Importance)} * \text{stddev(Importance)}))$$

The **higher the value of the IOppp indicator, the better is the opportunity** for improvement. Table 6 shows rankings based on the IOpp indicator where rank “1” indicates the **best** opportunity to improve either the targeted practice or the information to

**Table 6.** Ranking the Opportunities to Improve the Targeted Practice

Targeted Practice	Practice Ranking	Practice IOpp	Information Ranking	Information IOpp	Combined Rank
Sizing	3 (2)	0.67	3 (1)	0.52	3 (1)
Estimation	1 (3)	0.73	2 (4)	0.61	1 (2)
Scheduling	2 (1)	0.69	1 (2)	0.64	2 (3)
Task Sequencing	4 (5)	0.56	5 (5)	0.3	4 (4)
Task assignment	5 (4)	0.45	4 (3)	0.41	5 (5)

support people carrying out the targeted practice. Both practice and information rankings need to be considered, since the best opportunities are where is an opportunity to improve a targeted practice AND improve the information support for that practice. A Combined Rank is produced by pairing the opportunity ranking for each targeted practice with the opportunity ranking for its corresponding information. *The 2003 rankings are shown in brackets.*

### 3.8 Threats to the Validity of the 2006 Study

**Impact of the Respondents' Personal Characteristics:** Work experience, in general, and managerial experience and student ability, in particular, may have affected the respondents' perceptions of the information that they were given. Using parametric and nonparametric analysis, no significant correlations were found between a respondent's mean ratings for each generic IQ dimension and any of their personal characteristics. Analysis of variance found no significant difference between the undergraduate and postgraduate students in their mean ratings for each generic IQ dimension, apart from two items for which there is no obvious explanation. With 25 test items, it is most likely to be due to random effects.

**Non-Respondents:** Forty-seven percent of the students did not respond to the request to complete the survey even though a bonus mark was offered. If they constituted a set of students with different characteristics then the survey results could be compromised. There was no significant correlation between being a respondent and whether a student was an undergraduate or post-graduate. Within the set of undergraduate students, there was a significant association between subject marks and whether the student was a respondent with the better scoring students more likely to respond. This effect was absent from the PostGrad set. Analysis of variance between the undergraduate students' results on the two major assignments and the exam and whether they responded suggests that the set of respondents were more likely to be the better performing students. The Total\*Respondent analysis showed a highly significant association. However, the effect of ability on whether someone responded is relatively small (3%) and is unlikely to significantly affect the responses provided.

**Research Setting:** The project management tasks that the students performed were not situated within a real organisation with real goals. The students were on a steep learning curve and may have been unfamiliar with the technical language used in the instrument used to assess measurement. Runeson [14] has reported significant differences in performance on tasks between first-year undergraduate and graduate students undertaking a PSP course, raising concerns that even greater differences may exist between undergraduate students and practitioners. The likely threat to validity is that student respondents would be less able than practitioners to evaluate the quality of the information they were given. The conditions suggested by Tichy [17] with respect to student subjects applied in this study. Moreover, applying a cost-benefit approach [5] made it clear that the benefits of using students outweighed the costs. By using a student assignment it was possible to collect evaluations of the same information product. Thus, it was possible to control for the major independent variable since the same set of information was used by all – this is a degree of control that is impossible to achieve in a field study with practitioners.

### 3.9 Feedback on the Assessment Instrument

The assessment instrument was also subjected to evaluation so that incremental improvements can be made to the instrument with the goal of eventually producing an instrument of industrial quality. This evaluation consisted of objective measures of response times and subjective feedback from the respondents. The respondents were essentially positive in their feedback on the instrument and about their participation in the study. There were criticisms of the instrument: *the layout & colors used here are visually offensive :)* and *the help looked too long so I didn't read it. Hope these comments help improve the survey in the future at least. Cheers!* And, *How about just asking what was wrong? The survey seemed ambiguous and repetitive.* These positive comments were also received in the feedback on the goals of the survey: *Yes, it worth !! And I wish someone had performed a similar study \*last\* semester.*

A timestamp was placed on each response when the “submit” button was pressed. By tracking all the responses for a respondent, an indicator of the response time to make each set of responses was calculated. This is not a true measure of how long the respondent spent on making their responses; there may have been network delays or the respondent may have been involved in other activities. But by excluding outlier values and then calculating the mean time and variance for the remaining set of responses across the respondent sample, it was possible to estimate the maximum time to complete the survey with some confidence. It also enabled identification of those questions where respondents experienced difficulty

Subjects also provided feedback on the value and relevance of the assessment, the usability of the instrument and the comprehensiveness of the items in the instrument. Their responses were measured on the same 13 point scale used in the rest of the assessment. The respondents could also enter a free text comment.

## 4 The 2007 IQ Study: Evaluating the Improvements

The goal was to determine if the improvements made following the 2006 study had been effective. These improvements were a change to the evaluation instrument and improvements to the information provided to the 2007 student cohort. The analyses of the feedback obtained from the 2007 cohort were expected to show if the improvements had been effective. The assumption was that the 2006 cohort and 2007 cohort were sufficiently similar that differences in their responses were due to the improvements. To the extent that the course material and faculty members were the same, this assumption may have been justified, although experience suggests that student cohorts can vary considerably over years presenting a threat to the validity of longitudinal studies.

### Improvement to the Instrument: Revise the Interpretability Test Items

The low Alpha value for the Interpretability Test Items in the 2006 study suggests that the results for this dimension may not be reliable. The high number of respondents from non English-speaking backgrounds may also have caused them to view the word “interpret” from a linguistic perspective. The AIMQ definition for this construct is *“The extent to which information is in appropriate languages, symbols and units and the definitions are clear”*. Four of the five items were revised for the 2007 Study; for

example, “*It was easy to interpret what the information meant*” became “*There were clear definitions of the objects and concepts that were discussed in the information*”. The expectation was that the reliability (alpha) of the “Interpretability” test items would have improved (increased). However the alpha value dropped from 0.69 to 0.62. Ranking the means of the respondents’ ratings (Table 7) suggests that the “Interpretability” IQ dimension is no longer a significant issue based on the revised test items. “Ease of Use” and “Understandability” now stand out as the two dimensions of the information that require improvement.

**Table 7.** 2007 Assessment using the Generic IQ Model

IQ Dimension	Rank (2007)	Mean (2007)	Std. Dev (2007)	Rank (2006)	Rank (2003)
Ease of Use	1	6.3	2.62	1	=1
Understandability	2	6.3	2.76	3	4
Appropriate Amount	3	5.9	2.72	5	5
Interpretability	4	5.9	2.78	2	=1
Completeness	5	5.8	2.49	4	3
Relevancy	6	4.9	2.37	6	6

### **Improvement to the Information Product. Improve Information for Estimation**

The assignment required the students to estimate the effort and duration of the project. They first had to identify the tasks that were required to design, construct, test and implement the system described in the functional specification. Then they had to estimate the effort required to perform each task and transform this effort into duration. The final step was to plot the tasks on a PERT chart.

To assist them with estimating the effort required, they were provided with a Function Point Analysis of the proposed system showing the Logical Files, the Transactions and an unadjusted Function Point Count. They were also provided with a mean productivity rate for Java development based on the ISBSG database<sup>6</sup> and likely, pessimistic and optimistic estimates for the total effort to develop the system. They were told that the previous planning and specification phases had probably used about 22% of the project effort. After they had estimated the remaining effort by phase by summing the estimates for the tasks, they could check their phase estimates and project productivity rate against the industry norms in the ISBSG database using an online service. If necessary, they could then revise their estimates and re-check. The expectation was that the quality rating would have improved for the information to support:

1. ***Establish Estimates of Work Product and Task Attributes*** (CMMI Project Management SP 1.2-1).
2. ***Determine Estimates of Effort and Cost*** (CMMI Project Management SP 1.4-1).

<sup>6</sup> See <http://www.isbsg.org/>

The results of the assessment are presented in Table 8 where the mean scores for Practice Performance and Importance and Information Quality and Importance are listed. In Table 9 the Improvement Indicators (IOpp) that take into account both performance and importance are shown. Using these to rank the improvement opportunities shows that whereas “Estimation” was the best improvement opportunity in 2006; it was the third best opportunity in 2007, which suggests that the changes had been effective.

**Table 8.** 2007 Assessment using the Targeted IQ Model

	Practice		Information	
	Performance	Importance	Quality	Importance
<b>Targeted Practice</b>	<b>MeanR<sup>P</sup></b>	<b>MeanR<sup>PI</sup></b>	<b>MeanR<sup>M</sup></b>	<b>MeanR<sup>MI</sup></b>
Sizing	5.7	3.9	5.8	3.9
Estimation	6.1	4.1	5.2	4.3
Scheduling	5.8	3.1	5.0	3.9
Task sequencing	5.7	3.8	5.1	4.3
Task assignment	5.8	4.4	5.6	3.7

**Table 9.** 2007 Improvement Opportunities

Targeted Practice	Practice IOpp	Practice Rank	Information IOpp	Information Rank	Combined Rank (2007)	Combined Rank (2006)
Sizing	0.54	4	0.56	1	2	3
Estimation	0.61	2	0.25	4	3	1
Scheduling	0.79	1	0.30	3	1	2
Task Sequencing	0.58	3	0.21	5	4	4
Task assignment	0.36	5	0.55	2	5	5

## 5 Conclusions

The studies have demonstrated that it is possible to identify opportunities for improvement with instruments based on both a generic model of information quality (AIMQ [11]) and a targeted model; although assessment using the generic model does not clearly identify what must be changed.. Based on the 2007 study, there were improvements in the “Interpretability” dimension but it was unclear whether this was a result of the changes to the test items as it is possible that these items were measuring a different construct in 2007. By contrast, using a targeted model produced a highly specific and useful instrument to collect data that can be analysed to identify opportunities to improve both the information product and the targeted practice. The 2007 feedback showed that the improvements to the information given to the students had been effective with “Scheduling” and “Sizing” becoming the two practices now presenting the best improvement opportunities.

The study, so far, has demonstrated that the quality of information for software project management can be objectively assessed using both the generic model and the targeted model. The instrument that implements the generic model can be expanded to assess additional IQ dimensions with little effort. With more effort the targeted instrument could be expanded to cover the information needs of additional specific practices of project management. This longitudinal study using students of software project management is expected to continue while it is providing benefits to education and research. The next study will focus on the students' cognitive styles. Since "cognitive style" characterises how an individual prefers to search for and work with information, it may be a significant confounding variable in evaluations of information quality.

In terms of technology transfer, it is expected that further studies with practitioners will be necessary to demonstrate the validity of these models and methods for the formative assessment of the information provided to software project managers. In view of the difficulty that has been experienced in convincing the software engineering community that measurement and analysis is essential, it is unlikely that assessment of software measurement and analysis (meta-measurement) will become standard practice without very convincing evidence.

## References

1. Baddo, N., Hall, T., Wilson, D.: Implementing a people focused SPI programme. In: Proc 11th European Software Control and Metrics Conference Munich, April 2000 (2000)
2. Berry, M., Jeffery, D.R., Aurum, A.: Assessment of Software Measurement: an Information Quality Study. In: Proceedings of the Tenth International Software Metrics Symposium, IEEE Computer Society, Los Alamos (2004)
3. Berry, M., Vandenbroek, M.: A Targeted Assessment of the Software Measurement Process. In: Proceedings of the Seventh International Software Metrics Symposium, IEEE Computer Society, Los Alamos (2001)
4. Berry, M.: Assessment of software measurement. Australian Digital Theses - University of New South Wales (2006)
5. Carver, J., Jaccheri, L., Morasca, S., Shull, F.: Issues in using students in empirical studies in software engineering education. In: Proc. 9th International Software Metrics Symposium, September 3-5 (2003)
6. Eppler, M.J.: The Concept of Information Quality: An Interdisciplinary Evaluation of Recent Information Quality Frameworks. *Studies in Communication Sciences* 1, 167-182 (2001)
7. Frederiksen, H.D., Mathiassen, L.: Information-centric assessment of software metrics practices. *IEEE Transactions on Engineering Management* 52(3), 350-362 (2005)
8. ISO/IEC. ISO/IEC 15939: Information Technology -Software Measurement Process. 15939. Geneva, International Standards Organisation (2002)
9. Kahn, B.K., Strong, D.M., Wang, R.Y.: Information quality benchmarks: product and service performance. *Communications of the ACM* 45 (April 2002)
10. Lee, Y.W., Strong, D.: Process Knowledge and Data Quality Outcomes. In: Proc. Eighth International Conf. on Information Quality (ICIQ 2003), Boston (November 2003)
11. Lee, Y.W., Strong, D., Kahn, B.K., Wang, R.Y.: AIMQ: a methodology for information quality assessment. *Information and Management* 40, 133-146 (2002)
12. Price, R., Shanks, G.: A Semiotic Information Quality Framework. In: Proc 2004 IFIP Int. Conf. on Decision Support (2004)



13. Rivet, M.: A cognitive approach and the implementation of a measurement programme. In: Proc 11th European Software Control and Metrics Conf., Munich (April 2000)
14. Runeson, P.: Using Students as Experiment Subjects – An Analysis on Graduate and Freshman Student Data. In: Proc 7th Int. Conf. on Empirical Assessment & Evaluation in Software Engineering (April 2003)
15. SEI. CMMI for Systems Engineering/Software Engineering/Integrated Product and Process Development, Version 1.1, Continuous Representation (CMMI-SE/SW/IPPD, V1.1, Continuous), 12-1-2001. Pittsburgh, USA, Software Engineering Institute, Carnegie Mellon University (2001)
16. Tichy, W.F.: Hints for Reviewing Empirical Work in Software Engineering. Empirical Software Engineering Journal 5(4), 309–312 (2000)

## Appendix A: Generic IQ Instrument

Construct / Item Id	Test Item	Negative Coding
Relevancy		
10	The information was relevant to our work.	
11	The information was appropriate for our work.	
16	The information was applicable to our work.	
4	The information was useful to our work.	
Understandability		
17	The information was easy to comprehend.	
24	The meaning of the information was easy to understand.	
5	The information was easy to understand.	
9	The meaning of the information was difficult to understand.	Yes
Appropriate Amount		
1	The information was of sufficient volume for our needs.	
13	The amount of information was sufficient for our needs.	
6	The amount of information met our needs.	
Completeness		
20	The information covered the needs of our tasks.	
21	The information had sufficient breadth and depth for our tasks.	

Construct / Item Id	Test Item	Negative Coding
	8 The information was sufficiently complete for our needs.	
Ease of Use		
	14 The information was difficult to aggregate.	Yes
	2 The information was easy to manipulate to meet our needs.	
	22 The information was difficult to manipulate to meet our needs.	Yes
	25 The information was easy to combine with other information.	
	7 The information was easy to aggregate.	
Interpretability		
	12 The information was difficult to interpret.	Yes
	15 The measurement units for the information were clear.	
	18 It was difficult to interpret the coded information.	Yes
	23 The information was easily interpreted.	
	3 It was easy to interpret what the information meant.	

## Appendix B: Targeted IQ Instrument

Construct / Item Id	Targeted Practice / Test Item
Sizing	<i>Establish Estimates of Work Product and Task Attributes</i>
	27 The information that you were given in order to estimate the size of work products was complete.
	32 Sizing of work products was accurately performed.
	33 It is important that work products are accurately sized.
	34 The methods for estimating the size of work products were effective.
	35 It is important that there are effective methods for estimating the size of work products.
Estimation	<i>Determine Estimates of Effort and Cost</i>
	28 The amount of information you were given in order to estimate the effort required to develop new work products met your needs.

Construct / Item Id	Targeted Practice / Test Item
36	Estimation of the effort to develop new work products was sufficiently accurate.
40	It is important that assignments to team members are based on their skills and experience.
44	Information about the accuracy of previous estimates was provided in order to improve the reliability of the estimates.
48	It is important that the accuracy of previous estimates is provided to people who are estimating new work products.
Scheduling	<b><i>Establish the Budget and Schedule</i></b>
29	The information that you were given in order to develop the schedule was easy to combine with the other information that you had.
37	There is a high level of confidence in the reliability of the schedule that was developed .
41	It is important to have a reliable schedule.
45	The information on previous projects that was provided was sufficiently complete for our needs when developing the schedule.
49	It is important that the information on past projects is complete when scheduling.
Task Sequencing	<b><i>Establish the Budget and Schedule</i></b>
30	It was easy to interpret what the information on task dependencies meant.
38	The task dependencies are accurately modelled in the project schedule.
42	It is important that the task dependencies are accurately modelled in the project schedule.
46	Information on task dependencies from previous projects was used effectively to develop the project schedule.
50	It is important to have information on task dependencies from previous projects.
Task assignment	<b><i>Plan for Project Resources</i></b>
31	The information on the required skills and experience for the project team members was sufficient for our needs.
39	Responsibility for software work products and tasks was assigned to project team members based on information about their skills and experience.

Construct / Item Id	Targeted Practice / Test Item
43	It is important that assignments to team members are based on their skills and experience.
47	The information that was supplied gave us a good understanding of the required skills and experience that would be needed for the project team members.
51	It is important there is a good understanding of the required skills and experience for project team members when assigning work products and tasks.

# Software Cost Estimation Models Using Radial Basis Function Neural Networks

Ali Idri<sup>1</sup>, Azeddine Zahi<sup>2</sup>, Emilia Mendes<sup>3</sup>, and Abdelali Zakrani<sup>1</sup>

<sup>1</sup> Department of Software Engineering, ENSIAS, Mohamed V University  
Rabat, Morocco

idri@ensias.ma, zakrani@gmail.com

<sup>2</sup> Department of Computer Science FST, Sidi Mohamed Ben Abdellah University  
Fez, Morocco

azahi@fst-usmba.ac.ma

<sup>3</sup> Computer Science Department, The University of Auckland, Private Bag 92019  
Auckland, New Zealand

emilia@cs.auckland.ac.nz

**Abstract.** Radial Basis Function Neural Networks (RBFN) have been recently studied due to their qualification as an universal function approximation. This paper investigates the use of RBF neural networks for software cost estimation. The focus of this study is on the design of these networks, especially their middle layer composed of receptive fields, using two clustering techniques: the C-means and the APC-III algorithms. A comparison between a RBFN using C-means and a RBFN using APC-III, in terms of estimates accuracy, is hence presented. This study uses the COCOMO'81 dataset and data on Web applications from the Tukutuku database.

**Keywords:** Software effort estimation, Neural Networks, predictive accuracy, Radial basis function neural networks.

## 1 Introduction

Estimating software development effort remains a complex problem, and one which continues to attract considerable research attention. Improving the accuracy of the effort estimation models available to project managers would facilitate more effective control of time and budgets during software development. In order to make accurate estimates and avoid gross mis-estimations, several cost estimation techniques have been proposed, grouped into two major categories: (1) parametric models, which are derived from the statistical or numerical analysis of historical projects data [2], and (2) non-parametric models, which are based on a set of artificial intelligence techniques such as artificial neural networks, analogy-based reasoning, regression trees, genetic algorithms and rule-based induction [3,8,19,20,21]. In this paper, we focus on non-parametric cost estimation models based on artificial neural networks, and in particular Radial Basis Function Neural Networks.

Based on biological receptive fields, Moody and Darken proposed a network architecture referred to as the Radial Basis Function Network (RBFN), which employs

local receptive fields to perform function mappings [16]. RBFN can be used for a wide range of applications, primarily because it can approximate any regular function [17]. A RBFN is a three-layer feedforward network consisting of one input layer, one middle-layer and an output layer. Fig. 1 illustrates a possible RBFN architecture configured for web development effort. The RBFN generates output (effort) by propagating the initial inputs (cost drivers) through the middle-layer to the final output layer. Each input neuron corresponds to a component of an input vector. The middle layer contains  $M$  neurons, plus, eventually, one bias neuron. Each input neuron is fully connected to the middle-layer neurons. The activation function of each middle neuron is usually the Gaussian function:

$$f(x) = e^{-\left(\frac{\|x-c_i\|^2}{\sigma_i^2}\right)} \quad (1)$$

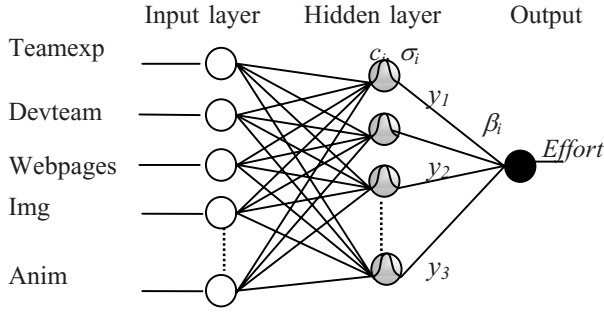
where  $c_i$  and  $\sigma_i$  are the center and the width of the  $i^{th}$  middle neuron respectively.  $\|\cdot\|$  denotes the Euclidean distance. Hence, each  $i^{th}$  hidden neuron has its own *receptive field* in the input space, a region centered on  $c_i$  with size proportional to  $\sigma_i$ . The Gaussian function decreases rapidly if the width  $\sigma_i$  is small, and slowly if it is large. The output layer consists of one output neuron that computes the development effort as a linear weighted sum of the middle layer outputs as follows:

$$Effort = \sum_{j=1}^M y_j \beta_j \quad \text{with} \quad y_j = e^{-\left(\frac{\|x-c_j\|^2}{\sigma_j^2}\right)} \quad (2)$$

The use of an RBFN to estimate software development effort requires the determination of the middle-layer parameters (receptive fields) and the weights  $\beta_j$ . The choice of the receptive fields, especially their distribution in the input space is often critical to the successful performance of an RBF network [18]. In general, there are two primary sources of this knowledge:

- The use of some clustering techniques to analyze and find clusters in the training data. The results of this grouping are used to establish prototypes of the receptive fields.
- The use of existing empirical domain knowledge to form the set of receptive fields. Along this line emerge some fuzzy models; in particular, some interesting equivalence between RBF networks and Fuzzy Rule-Based systems [11].

The aim of this study is to discuss the preprocessing phase of the RBFN networks in designing software cost estimation as it occurs through data clustering techniques. More especially, we use two clustering techniques: 1) the APC-III algorithm developed by Hwang and Bang [6], and 2) the best-known clustering method that is the C-means algorithm. In an earlier work [10], we have empirically evaluated these two clustering techniques, when designing RBF neural networks for software cost estimation, using the COCOMO'81 dataset. The RBFN designed with the C-means algorithm performed better, in terms of cost estimates accuracy, than the RBFN designed with the APC-III algorithm [10]. To confirm those results, this paper replicates that study using a dataset of 53 Web projects from the Tukutuku database [14].



**Fig. 1.** An example of Radial Basis Function Network architecture for Web development effort

This paper is structured as follows. Section 2 briefly describes how the two clustering algorithms APC-III and C-means, can be used in the design of an RBF neural network. In Section 3, the empirical results obtained using an RBFN on the CO-COMO'81 and the Tukutuku datasets are discussed and compared in terms of prediction accuracy. Conclusions and an overview of future work are presented in Section 5.

## 2 Clustering Techniques for RBF Neural Networks

Clustering techniques have been successfully used in many application domains, including biology, medicine, economics, and patterns recognition [1,4,5,12,13]. These techniques can be grouped into major categories: Hierarchical or Partitional [5]. In this paper, we focus only on the Partitional clustering algorithm since it is used more frequently than other clustering algorithms in pattern recognition fields. Generally, Partitional clustering algorithms suppose that the data set can be well represented by finite prototypes. Partitional clustering is also referred to as an objective function-based clustering algorithm.

Clustering has been often used as a pre-processing phase in the design of RBF neural networks, where its primary aim is to set up an initial distribution of the receptive fields (hidden neurons) across the input variables space of the input variables. In particular, this implies a location of the modal values of these fields (e.g., the modes of the Gaussian functions).

In this work, we use and compare the C-means and the APC-III clustering algorithms to determine the receptive fields of an RBF network for software cost estimation [9,10]. These two clustering techniques are briefly presented next.

APC-III is a one-pass clustering algorithm with a constant radius  $R_0$  defined as follows:

$$R_0 = \alpha \frac{1}{N} \sum_{i=1}^N \min_{i \neq j} (\|P_i - P_j\|) \quad (3)$$

Where  $N$  is the number of historical software projects and  $\alpha$  is a predetermined constant.  $R_0$  expresses the radius of each cluster and consequently it controls the number of the clusters provided. APC-III generates many clusters if  $R_0$  is small and few clusters if it is large. The outline of the APC-III algorithm can be stated as follows:

- 1- Initially, the number of clusters is set at 1; the center of this cluster  $C_1$  is the first software project in the dataset, say  $P_1$
- 2- Iterate from  $i=2$  to  $N$  ( $N$  is the number of historical software projects)
  - a. For  $j=1$  to  $c$  ( $c$  is the number of clusters)
    - Compute  $d_{ij}$  ( $d_{ij}$  is the Euclidean distance of  $P_i$  and  $c_j$ ,  $c_j$  is the center of cluster  $C_j$ )
    - If  $d_{ij} < R_0$  then
      - Include  $P_i$  into  $C_j$  and adjust the center of  $C_j$
      - Exit from the loop
  - b. If  $P_i$  is not included in any clusters then
    - Create a new cluster that has  $P_i$  as a center

C-means clustering algorithm has been successfully applications in fields such as pattern recognition and data compression. It is a multi-pass and time-consuming clustering algorithm. The C-means algorithm partitions a collection of  $N$  vectors into  $c$  clusters  $C_i$ ,  $i=1, \dots, c$ . The aim is to find cluster centers (centroids) by minimizing a dissimilarity (or distance) function which is given in below.

$$J = \sum_{i=1}^c \sum_{x_k \in C_i} d(x_k, c_i) \quad (4)$$

where  $c_i$  is the center of cluster  $C_i$ ;  $d(x_k, c_i)$  is the distance between  $i^{th}$  center ( $c_i$ ) and  $k^{th}$  data point. For simplicity, the Euclidian distance is used as dissimilarity measure and the overall dissimilarity function is expressed as follows.

$$J = \sum_{i=1}^c \sum_{x_k \in C_i} \|x_k - c_i\|^2 \quad (5)$$

The outline of the C-means algorithm can be stated as follows:

- 1- Define the number of the desired clusters,  $c$ .
- 2- Initialize the centers  $C_i, i=1..c$ . This is typically achieved by randomly selecting  $c$  points from among all of the data points.
- 3- Compute the Euclidean distance between  $x_j$  and  $c_i$ ,  $j=1..N$  and  $i=1..c$
- 4- Assign each  $x_j$  to the most closer cluster  $C_i$
- 5- Recalculate the centers  $c_i$
- 6- Compute the objective function  $J$  given in Equation 5. Stop if its improvement over previous iteration is below a threshold.
- 7- Iterate from step 3.

As mentioned earlier, the use of an RBFN to estimate software development effort requires the determination of its architecture parameters according to the characteristics of the used datasets (COCOMO'81 or Tukutuku database), especially the number of input neurons, number of hidden neurons, centers  $c_i$ , widths  $\sigma_i$  and weights  $\beta_j$ .

The number of input neurons is the same as the number of attributes (cost drivers) describing the historical software projects in the dataset. Hence, the number of input neurons is equal to 13 for COCOMO'81 and is equal to 9 for the Tukutuku dataset. The number of hidden neurons is determined by the number of clusters ( $c$ ) provided



by the APC-III or the C-means algorithms described in Section 2. Concerning the widths  $\sigma_i$ , they were usually determined in the literature to cover the input space as uniformly as possible [6]. Covering the historical software project space uniformly implies that the RBFN will be able to generate an estimate for a new project even though it is not similar to any historical project. In such a situation, we prefer that the RBFN does not provide any estimate than one that may easily lead to wrong managerial decisions and project failure. In [9,10], we had adopted a simple strategy based primarily on assigning one value to all  $\sigma_i$ ; this value depends on the radius  $R_0$  used in the APC-III algorithm. Because here we investigate two clustering techniques, we adopt one of the following formulas to determine  $\sigma_i$ :

$$\sigma_i = \begin{cases} \max_{x_j \in C_i} d(x_j, c_i) & \text{if } \text{card}(C_i) > 1 \\ \max_{k / \text{card}(C_k) > 1} \sigma_k & \text{if } \text{card}(C_i) = 1 \end{cases} \quad (6)$$

$$\sigma_i = \begin{cases} \max_{x_j \in C_i} d(x_j, c_i) & \text{if } \text{card}(C_i) > 1 \\ \min_{k / \text{card}(C_k) > 1} \sigma_k & \text{if } \text{card}(C_i) = 1 \end{cases} \quad (7)$$

where  $\text{card}(C_i)$  is the cardinality of cluster  $C_i$ .

Concerning the weights  $\beta_j$ , we may set each  $\beta_j$  to the associated effort of the center of the  $j^{\text{th}}$  neuron. However, this technique is not optimal and does not take into account the overlapping that may exist between receptive fields of the hidden layer. Thus, we use the Delta rule to derive the values of  $\beta_j$ .

### 3 Data Description

In this work, we evaluate empirically these two clustering techniques when designing RBF neural networks for software cost estimation based on two different databases: The COCOMO'81 dataset and data web from Tukuruku database.

- The COCOMO'81 dataset contains 252 software projects which are mostly scientific applications developed by Fortran [2,10]; Each project is described by 13 attributes (see Table 1) : the software size measured in KDSI (Kilo Delivered Source Instructions) and the remaining 12 attributes are measured on a scale composed of six linguistic values: 'very low', 'low', 'nominal', 'high', 'very high' and 'extra high'. These 12 attributes are related to the software development environment such as the experience of the personnel involved in the software project, the method used in the development and the time and storage constraints imposed on the software.

- The Tukuruku dataset contains 53 Web projects [14,15]. Each Web application is described using 9 numerical attributes such as: the number of html or shtml files used, the number of media files and team experience (see Table 2). However, each project volunteered to the Tukuruku database was initially characterized using more than 9 software attributes, but some of them were grouped together. For example, we grouped together the following three attributes: number of new Web pages developed by the team, number of Web pages provided by the customer and the number of Web pages developed by a third party (outsourced) in one attribute reflecting the total number of Web pages in the application (Webpages).

**Table 1.** Software attributes for COCOMO'81

Software attribute	Description
SIZE	Software Size
DATA	Database Size
TIME	Execution Time Constraint
STOR	Main Storage Constraint
VIRTMIN	Virtual Machine Volatility
VIRT MAJ	Virtual Machine Volatility
TURN	Computer Turnaround
ACAP	Analyst Capability
AEXP	Applications Experience
PCAP	Programmer Capability
VEXP	Virtual Machine Experience
LEXP	Programming Language Experience
SCED	Required Development

**Table 2.** Software attributes for the web dataset

Attribute	Description
Teamexp	Average number of years of experience the team has on Web development
Devteam	Number of people who worked on the software project
Webpages	Number of web pages in the application
Textpages	Number of text pages in the application(text page has 600 words)
Img	Number of images in the application
Anim	Number of animations in the application
Audio/video	Number of audio/video files in the application
Tot-high	Number of high effort features in the application
Tot-nhigh	Number of low effort features in the application

## 4 Empirical Results

The following section presents and discusses the results obtained when applying the RBFN to the COCOMO'81 and the Tukutuku datasets. The calculations were made using two software prototypes developed using the C programming language under a Microsoft Windows PC environment. The first software prototype implements the APC-III and the C-means clustering algorithms, providing both the clusters and their centers from the database. The second software prototype implements a cost estimation model based on an RBFN architecture in which the middle-layer parameters are determined by means of the first software prototype.

The accuracy of the estimates generated by the RBFN is evaluated by means of the Mean Magnitude of Relative Error MMRE and the Prediction level Pred defined as follows:

$$MMRE = \frac{1}{N} \sum_{i=1}^N \left| \frac{Effort_{actual,i} - Effort_{estimated,i}}{Effort_{actual,i}} \right| \times 100 \quad (8)$$

$$Pred(p) = \frac{k}{N} \quad (9)$$

where  $N$  is the total number of observations,  $k$  is the number of observations with an MRE less than or equal to  $p$ . A common value for  $p$  is 25.

To decide on the number of hidden units, we have conducted several experiments with both the C-means and the APC-III algorithms. These experiments use the full dataset for training. Choosing the ‘best’ classification to determine the number of hidden neurons and their centers is a complex task. For software cost estimation, we suggest that the ‘best’ classification is the one that satisfies the following criteria:

- It provides coherent clusters, i.e. the software projects of a given cluster have satisfactory degrees of similarity;
- It improves the accuracy of the RBFN.

where  $N$  is the total number of observations,  $k$  is the number of observations with an MRE less than or equal to  $p$ . A common value for  $p$  is 25.

#### 4.1 RBFN with C-Means Algorithm

To measure the coherence of clusters in the case of the C-means algorithm, one of the two criteria was used: the objective function  $J$  given in Equation 5 or the Dunn’s index defined by the following Equation:

$$D_1 = \min_{1 \leq i \leq c} \left( \min_{i+1 \leq j \leq c-1} \left( \frac{d(C_i, C_j)}{\max_{1 \leq k \leq c} (d(C_k))} \right) \right) \quad (10)$$

$$d(C_i, C_j) = \min_{x_i \in C_i, x_j \in C_j} d(x_i, x_j) \quad d(C_k) = \max_{x_i, x_j \in C_k} d(x_i, x_j)$$

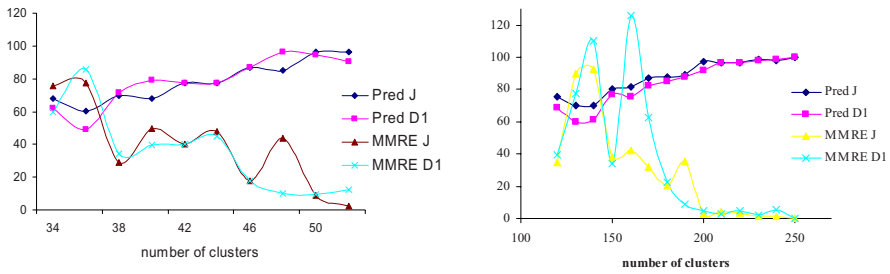
where  $d(C_i, C_j)$  is the distance between clusters  $C_i$  and  $C_j$  (intercluster distance); and  $d(C_k)$  is the intracluster distance of cluster  $C_k$ .

The Dunn’s index ( $D_1$ ) expresses the idea of identifying clusters that are compact and well separated. The main goal of the measure ( $D_1$ ) is to maximize the intercluster distances and minimize the intra-cluster distances. Therefore, the number of cluster that maximizes  $D_1$  is taken as the optimal number of the clusters.

Several experiments were conducted with an RBFN, each time using one of the classifications generated by the C-means algorithm. These experiments used the full set of historical projects for training and testing. For each number of clusters ( $c$ ), the RBFN used the two C-means classifications that respectively minimise  $J$  or maximise  $D_1$ . For instance, when fixing  $c$  to 34 for the Tuketuku dataset, the two chosen classifications were respectively those for which  $J$  is equal to 9299.78 and  $D_1$  is equal to 7.14.

Fig. 2-a and Fig 2-b show the relationship between the accuracy of the RBFN on the Tuketuku and the COCOMO’81 datasets, measured in terms of  $Pred(0.25)$  and

MMRE, and the used classifications (number of clusters) minimizing the objective function  $J$  or maximizing the  $D_I$  index. It can be noticed for both datasets that the accuracy of the RBFN when using the C-means classification that minimizes  $J$  (Pred\_J and MMRE\_J) is superior to the accuracy obtained when using the classification maximizing  $D_I$  (Pred\_D1 and MMRE\_D1). Fig. 2-a (resp. Figure 2-b) shows only the results of experiments for the Tukutuku dataset (resp. the COCOMO'81 dataset) when the number of clusters is higher than 34 (resp. is higher than 120) because the evaluated accuracy of the RBFN is acceptable (the common values used in the literature are  $\text{Pred}(25) \geq 70$  and  $\text{MMRE} \leq 30$ ). Also, the obtained classifications for  $c$  lower than 34 (resp. lower than 120) are, in general, less coherent, i.e. some clusters are composed of projects that are not sufficiently similar; for those projects, the RBFN may generate inaccurate estimates.



(a) Based on Tukutuku dataset

(b) Based on COCOMO'81 dataset

**Fig. 2.** Relationship between the accuracy of RBFN (MMRE and Pred) and the used classification of the C-means (J and D1) for the Tukutuku and the COCOMO'81 dataset

## 4.2 RBFN with the APC-III Algorithm

The classifications generated by the APC-III algorithm depend on the number  $\alpha$  that defines the radius  $R_0$ . Table 3 shows the classifications obtained according to different values of  $\alpha$  (see Equation 3).

The results presented in Table 3-a and Table 3-b show that the number of clusters is monotonic decreasing according to  $\alpha$ . This is due to the fact that the radius  $R_0$  is monotonic increasing according to  $\alpha$ . For each value of  $\alpha$ , the presentation sequence of the software projects was varied. Indeed, the classification provided by the APC-III depends on this presentation sequence because it influences the determination of the centers; for each  $\alpha$  we retained the classification minimizing the number of clusters.

Thus, several experiments for each dataset were conducted using an RBFN, each time employing one of the classifications of the Table 3-a or Table 3-b. These experiments use the entire dataset of projects for training and testing. The weights  $\beta_j$  were calculated using the Delta learning rule. The RBFN converges quickly for both datasets, with fewer than 12000 iterations of learning. The accuracy of the estimates generated by the RBFN was evaluated by means of the MMRE and the Pred(25) indicators. Fig. 3-a corresponding to the Tukutuku dataset shows the MMRE and Pred as functions of the

**Table 3.** Number of clusters according to  $\alpha$  for the Tuketuku and the COCOMO'81 datasets

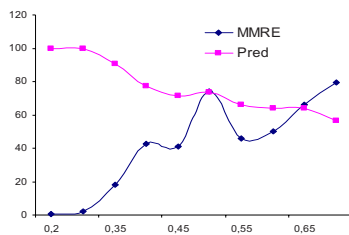
$\alpha$	Number of clusters
0.2 0.25 0.35 0.4 0.45	52  50  48  44  43
0.5 0.55 0.6	41  38 37
0.65 0.7	36 34

(a) For the Tuketuku dataset

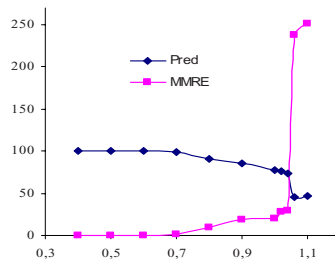
$\alpha$	Number of clusters
0.4 0.5 0.6 0.7 0.8	251 244 234 216 200
0.9 1.0 1.02 1.04	189 170 162 161
1.06 1.08 1.1	155 151 150

(b) For COCOMO'81 dataset

classification ( $\alpha$ ). It can be noticed that the accuracy of the RBFN is better when  $\alpha$  is lower than 0.5 (MMRE=73.88 and Pred(25)=73.58). When  $\alpha$  is higher than 0.5, the MMRE and pred(25) become not acceptable. Fig. 3-b corresponding to the COCOMO'81 dataset shows the MMRE and Pred as functions of the classification ( $\alpha$ ). It can be noticed that the accuracy of the RBFN is better when  $\alpha$  is lower than 1.04 (MMRE=29.81 and Pred(25)=73.81). When  $\alpha$  is higher than 0.5, the MMRE and pred(25) become not acceptable.



(a) Based on Tuketuku dataset

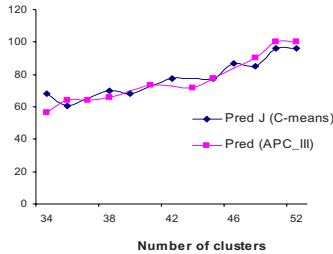


(b) based on COCOMO'81 dataset dataset

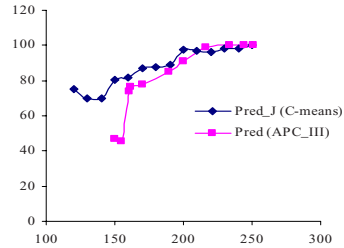
**Fig. 3.** Relationship between the accuracy of RBFN (MMRE and Pred) and the used classification of the APC-III ( $\alpha$ ) for the Tuketuku and the COCOMO'81 datasets

To conclude this Section, the accuracy of the RBFN using the C-means algorithm is compared with that of the RBFN when using the APC-III algorithm for the Tuketuku dataset (Fig. 4-a) and the COCOMO'81 dataset (Fig. 4-b). It can be noticed that the RBFN with C-means performs better than the RBFN with APC-III for both datasets. Indeed, for the Tuketuku dataset, an acceptable accuracy of the RBFN-C-Means is still achieved until the number of clusters is equal to 34; by contrast, it was acceptable only

until the number of clusters is equal to 41 in the case of the RBFN-APC-III. For the CO-COMO'81 dataset, an acceptable accuracy of the RBFN-C-Means is still achieved until the number of clusters is equal to 120; by contrast, it was acceptable only until the number of clusters is equal to 150 in the case of the RBFN-APC-III.



(a) Based on Web hypermedia applications



(b) Based on COCO MO'81

**Fig. 4.** Comparing the accuracy of RBFN using C-means (Pred\_J) and RBFN using APC-III (Pred)

## 5 Conclusions and Comments on Future Work

In this paper, we have empirically studied the use of two clustering techniques when designing RBF neural networks for software cost estimation. The two used clustering algorithms are the well-known C-means and the APC-III. This comparative study is based on the COCOMO'81 dataset and a web hypermedia applications dataset that contains 53 software projects. We used the entire dataset to train and test the designed RBFN. We have found that the RBFN designed with the C-means algorithm performs better, in terms of cost estimates accuracy, than the RBFN designed with the APC-III algorithm for the two datasets. Hence, we recommended the use of C-means algorithm for building software cost estimation models based on RBF neural networks. To deal with imprecision and uncertainty in the cost estimation process based on RBF neural networks, we are working currently in designing an RBFN based on Fuzzy C-means.

## References

- [1] Berkhin, P.: Survey Of Clustering Data Mining Techniques (2002), <http://citeseer.nj.nec.com/berkhin02survey.html>
- [2] Boehm, B.W.: Software Engineering Economics. Prentice-Hall (1981)
- [3] Burgess, C.J., Lefley, M.: Can Genetic Programming Improve Software Effort Estimation? *Information and Software Technology* 43, 863–873 (2001)
- [4] Duda, R.O., Hart, P.E.: Pattern Classification and Scheme Analysis. Wiley, New York (1973)
- [5] Dunham, M.H.: Datamining: Introduction and Advanced Topics. Prentice-Hall (2003)
- [6] Hwang, Y.-S., Bang, S.-Y.: An Efficient Method to Construct a Radial Basis Function Network Classifier. *Neural Networks* 10(8), 1495–1503 (1997)

- [7] Idri, A., Khoshgoftaar, T.M., Abran, A.: Can Neural Networks be easily Interpreted in Software Cost Estimation. In: FUZZ-IEEE, Hawaii, pp. 1162–1166 (2002)
- [8] Idri, A., Abran, A., Khoshgoftaar, T.M.: Estimating Software Project Effort by Analogy based on Linguistic values. In: 8th IEEE International Software Metrics Symposium, Ottawa, Canada, April 4–7, pp. 21–30 (2002)
- [9] Idri, A., Abran, A., Mbarki, S.: Validating and Understanding Software Cost Estimation Models based on Neural Networks. In: International Conference on Information and Communication Technologies: from Theory to Applications (ICTTA), Damascus, Syria, April 19–23, pp. 433–438 (2004)
- [10] Idri, A., Abran, A., Mbarki, S.: An Experiment on the Design of Radial Basis Function Neural Networks for Software Cost Estimation. In: International Conference on Information and Communication Technologies: from Theory to Applications (ICTTA), Damascus, Syria, April 24–28, pp. 433–438 (2006)
- [11] Jang, J.S., Sun, C.T.: Functional equivalence between radial basis function networks and fuzzy inference systems. *IEEE Trans. on Neural Networks* 4, 156–158 (1992)
- [12] Yu, J.: General C-Means Clustering Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27(8) (August 2005)
- [13] Kaufman, L., Rousseeuw, P.J.: *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, Inc., New York (1990)
- [14] Kitchenham, B.A., Mendes, E.: A Comparison of Cross-company and Within-company Effort Estimation Models for Web Applications. In: *Proceedings of EASE Conference*, pp. 47–56 (2004)
- [15] Mendes, E., Triggs, W.C., Mosley, N., Counsell, S.: A comparison of Development Effort Estimation Techniques for Web Hypermedia Applications. In: 8th IEEE International Software Metrics Symposium, Ottawa, pp. 131–140 (2002)
- [16] Moody, J., Darken, C.: Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation* 1, 281–294 (1989)
- [17] Park, J., Sandberg, I.W.: Approximation and Radial Basis Function Networks. *Neural Computation* 5, 305–316 (1993)
- [18] Pedrycs, W.: Conditionnal Fuzzy Clustering in the Design of Radial Basis Function Neural Networks. *IEEE Transaction on Neural Networks* 9(4) (July 1998)
- [19] Shepperd, M., Schofield, C.: Estimating Software Project Effort Using Analogies. *Transactions on Software Engineering* 23(12), 736–747 (1997)
- [20] Srinivasan, K., Fisher, D.: Machine Learning Approaches to Estimating Software Development Effort. *IEEE Transactions on Software Engineering* 21(2), 126–136 (1995)
- [21] Wittig, G., Finnie, G.: Estimating Software Development Effort with Connectionist Models. *Information and Software Technology* 39, 469–476 (1997)

# Towards an Early Usability Evaluation for Web Applications\*

Jose Ignacio Panach, Nelly Condori-Fernández, Francisco Valverde,  
Nathalie Aquino, and Oscar Pastor

Department of Information Systems and Computation  
Technical University of Valencia  
Camino de Vera s/n, 46022 Valencia, Spain  
{jpanach,nelly,fvalverde,naquino,opastor}@dsic.upv.es  
Phone: +34 96 387 7000, Fax: +34 96 3877359

**Abstract.** In the Human-Computer Interaction (HCI) community, the usual way to measure usability is through a user test. The disadvantage of this way is that the system must be implemented before performing the test. As a consequence, developers must solve the usability issues in the last stages of the development process. Currently, the model-driven software development is gaining popularity as a solution to reduce changes impact. In this paper, a usability model is proposed to evaluate early usability from the conceptual schemas that represents a Web Application. This approach allows to incorporate usability improvements before the implementation of the final web application. We evaluate the usability of artefacts modelled with OOWS, a model-driven web engineering method. In addition, two case studies are used to verify the instruments proposed to evaluate our early usability model.

**Keywords:** Usability, conceptual models, metrics, usability patterns, Web engineering methods, automatic code generation.

## 1 Introduction

According to ISO/IEC 9126-1 [12], usability is one of the characteristics that define Software Quality together with Functionality, Reliability, Efficiency, Maintainability, and Portability. The user can reject a system with poor usability even though functionality is adequate. Usability is therefore an essential software characteristic.

In order to decide whether a system is usable or not, metrics need to be defined to measure system usability. In the past, most of these measurements have been carried out in the final system [27][8] by means of user testing. However, with this approach, changes aiming at improving the system are difficult to apply because they affect several phases throughout the software development process. Moreover, user testing requires too much time and effort to carry it out, so few software systems can evaluate their usability. For this reason we propose an early usability evaluation, on the basis

---

\* This work has been developed with the support of MEC under the project SESAMO TIN2007-62894.

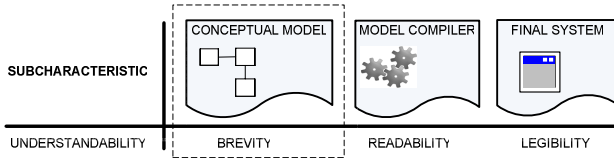


of an existing generic usability model [7][1]. This paper uses the usability model in order to evaluate the entire OOWS generation process [9], a development method that allows automatic generation of Web applications from a Conceptual Model through a Model Compiler. The proposed usability evaluation is based on the conceptual primitives that represent the system.

The usability model includes a set of sub-characteristics that are composed of several attributes. The attributes can be measured at three levels: conceptual model, model compiler, and final system.

- The attributes from the generated system are measured with questionnaires that are completed by the users. These attributes represent subjective usability aspects.
- The attributes that measure usability aspects inserted by the Model Compiler are applied in each generated system in the same way. These attributes measure usability aspects that do not depend on the analyst choices; they only depend on a decision of the Model Compiler. Therefore, it has not sense to measure these attributes, because their values are always the same in all the systems.
- The attributes that can be measured in the Conceptual Model represent the measurable attributes in an *early evaluation*. The usability measure of these attributes depends on the way that the Conceptual Model is built.

This paper focuses on this last attributes group. Early usability evaluation proposed includes some attributes that affects only the Conceptual Model. Figure 1 shows as example that *Understandability* sub-characteristic is composed of *Brevity*, *Readability* and *Legibility* attributes, and each of these attributes is measurable at a different level.



**Fig. 1.** Decomposition of sub-characteristics in attributes

The contributions of this paper are:

- (1) to detect measurable attributes in the Conceptual Model;
- (2) to define metrics to measure the usability of the detected attributes;
- (3) to interpret the measures obtained;
- (4) to add together all the indicators to obtain the usability of the entire system.

The paper is structured as follows. In section 2, this paper presents a survey of other usability models proposed in the literature. Section 3 describes the web engineering method used for the early usability evaluation. Section 4 shows a proposal to measure usability in the Conceptual Model. Section 5 explains the a priori validation of the early usability model proposed. Finally, we discuss conclusions and future work.

## 2 Other Usability Models

In recent years, some usability models have been proposed in the Software Engineering (SE) community and in the Human-Computer Interaction (HCI) community. On the one hand, the SE community has defined a standard usability model called ISO/IEC 9126-1 [12]. On the other hand, in the HCI community, ISO 9241-11 [11] explains how usability can be specified in terms of user performance and satisfaction. Both points of view are general; therefore, these models must be made more specific in order to be able to use them in the usability measure of a specific system.

Some authors have proposed their own usability models, such as Dromey [6]. He defined two properties: *behaviours* and *uses*. Behaviour is something that the software exhibits when it is executed under the influence of a set of inputs. A use is what different interest groups do with the software. In the same way, Dix et al. [5] propose a usability model based on three main categories: *learnability*, *flexibility*, and *robustness*. This model starts from categories and arrives at specific principles. This approach is similar to the usability model that is proposed in this paper. In our proposed model, the categories are equivalent to sub-characteristics, and specific principles are equivalent to attributes.

The model proposed by Nielsen [20] is more detailed than the Dix model. It focuses on social and practical acceptability. Usability is considered to be a sub-characteristic of usefulness, which is, in turn, a sub-characteristic of practical acceptability. A similar model, which only differs from the Nielsen model in the terminology used, is proposed by Shneiderman [25]. Shneiderman defines usability as “five measurable human factors that are central to the evaluation of human factor goals”. These factors are: *speed of performance*, *learning time*, *retention over time*, *rate of errors by users*, and *subjective satisfaction*.

Moreover, there are other proposals based on standard extensions, like the Van Welie et al. [27] model. These researchers propose a model with three related layers: *Usability*, *Usage indicators*, and *Means*. A similar approach is proposed by Fitzpatrick [8], who defines three strands to identify the attributes of usable software: *Software quality*, *Statutory obligations*, and *Human-computer interaction*. The Van Welie and Fitzpatrick models are centred on user tests, ignoring an early evaluation such as the one proposed in our work.

Finally, some usability measures have been designed exclusively for Web environments. An example is the Web-site QEM methodology defined by Olsina [22]. This methodology is essentially quantitative, flexible, and robust, covering most of the activities in the evaluation, comparison, and ranking process of websites. However, this methodology is related to the final interface and cannot be used to measure usability in Conceptual Modelling.

In the next section we introduce OOWS, a web engineering method based on transformation models.

## 3 OOWS: A Development Method for Web Environment

OO-Method [23] is an object oriented software production method that generates information systems automatically. OO-Method models the system in different abstraction

levels, distinguishing between problem space (the most abstract level) and solution space (the lowest abstract level). The system is represented by a set of Conceptual Models: 1) A Class Diagram that represents the static structure; and 2) The State and Functional Diagrams that represents the behaviour. OOWS [9] is the OO-Method extension in order to model and generate Web Applications. The OOWS conceptual schema is defined from two models that describe the different concerns of a web application: the Navigational Model and the Presentation Model.

- **Navigational Model:** This model defines the system navigational structure. It describes the navigation allowed for each type of user by means of a Navigational Map (Fig. 2. right). This map is depicted by means of a directed graph whose nodes represent navigational contexts and their arcs represent navigational links that define the valid navigational paths over the system. Navigational contexts are made up of a set of **Abstract Information Units** (AIU) (Fig. 2. left), which represent the requirement of retrieving a chunk of related information. AIUs are views defined over the underlying OO-Method Class Diagram. These views are represented graphically as UML classes that are stereotyped with the «view» keyword and that contain the set of attributes and operations that will be available to the user. Basically, an AIU represents a web page of the Web Application at conceptual level. Figure 2 shows an example of AIU and a Navigational map.

Sometimes, the retrieved information is difficult to manage due to its size. To simplify the information browsing, two search mechanisms are provided: indexes and filters. Indexes create a list of summarized information using an attribute from the AIU. For instance, an index by year shows a list of years to the user. When the user selects a concrete year, only the books from that year are shown. Additionally, filters define a population condition to restrict the object instances to be retrieved depending on a value provided by the user. Thanks to filters, the user can retrieve only the books from a year inserted by him. These two search mechanisms are valuable from a usability point of view.

- **Presentation Model:** Thanks to this model we are able to specify the visual properties of the information to be shown. To achieve this goal, a set of presentation patterns are proposed to be applied over our conceptual primitives. Examples of presentation patterns are: (1) Information Layout: to arrange information, OOWS provide three basic layout patterns, register, tabular, and tree, and one layout operator, master-detail; (2) Pagination: this mechanism allows us to specify information by “scrolling”. All the retrieved instances are broken into logical blocks of a given cardinality; (3) Ordering Criteria: this pattern defines a class population ordering (ASCendant or DESCendant) using the value of one or more attributes.

Both models are complemented by OO-Method models that represents functional and persistence layers. OOWS generates the code corresponding to the user interaction layer, and OlivaNOVA [3], the industrial OO-Method implementation, generates the business logic and the persistence layers.

OOWS development process is compliant with Model Driven Architecture (MDA) principles [17], where a Model Compiler transforms a Platform Independent Model (PIM) into its corresponding Software Product. This MDA transformation process has been implemented by means of a case tool and a model compiler. Further details can be found in [26].

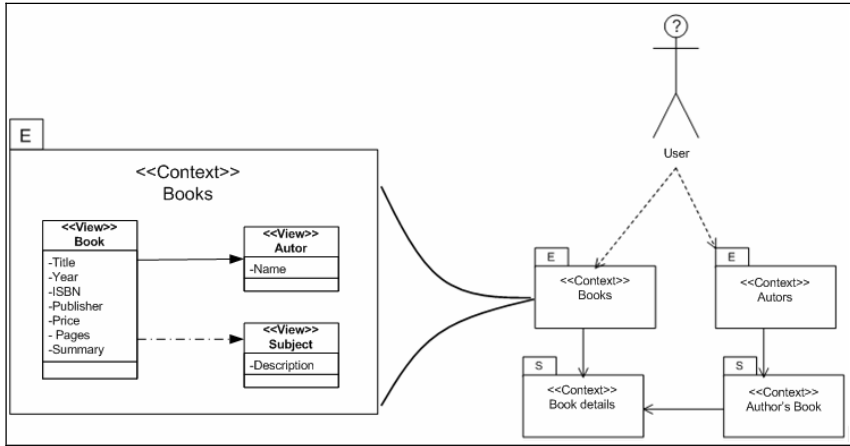


Fig. 2. Example of AIU and Navigational map

## 4 A Method for Measuring Usability in Software Systems

In previous work, a usability model based on the ISO 9126-1 [12] was defined to establish the aspects that make up system usability [7][1]. This usability model has two primitives: sub-characteristics and attributes. A *sub-characteristic* represents a general aspect related to usability while an *attribute* is a measurable aspect of the usability. Sub-characteristics are composed of several attributes. Therefore, the usability model has a hierarchical decomposition. This decomposition starts with sub-characteristics and ends with measurable attributes. Sub-characteristics were extracted from the sub-characteristic of the ISO 9126-1. Attributes were extracted from several usability taxonomies as Bastien and Scapin's work [2]. The proposed method for measuring usability is based on these measurable attributes of the usability model.

The process of defining an early evaluation includes: a set of *metrics* to assign a value to each usability attribute; a number of *indicators* to classify the generated metrics; and the usability *results* extracted from the indicator of each attribute.

Each metric assigns a quantitative value for each attribute from the usability model. The metrics for each attribute are the following:

1. Action determination: This attribute is used to help the user to understand the meaning of his/her future actions. This attribute is related to the learnability sub-characteristic. In OOWS, the specification of an alias (a large description) for a Filter, a Context and a Service corresponds to this attribute. Filter, Context and Services with alias helps to obtain a more usable system. In case of the analyst does not define an alias, the Model Compiler uses the identifier of the Filter, the Context or the Service as alias.
2. Labelling significance: This attribute is used to measure whether or not a label has significance and is related to the learnability sub-characteristic. In OOWS, this attribute can be measured counting the number of attributes with alias defined in the classes of the OO-Method Class Diagram. For the attributes without alias, the Model Compiler assigns the identifier of the attribute as alias.

3. Brevity: This attribute, which belongs to the understandability sub-characteristic, measures the cognitive effort of the user. Some studies have demonstrated that the human memory has the capacity to retain three different scenarios [15]. Therefore, each reachable context through more than three contexts decreases the usability.
4. Information density: This attribute, which is related to understandability, measures whether too much information is shown in an interface. It takes several measures into account:
  - The more attributes there are in a context view, the more information density there will be. Some Web guidelines [16] recommend writings with a line of characters between 75 and 100 characters in order to avoid using scroll. In other words, the context should include at most ten attributes (taking an average of seven characters per attribute).
  - The same criterion can be used for the number of services. If a context has more than seven services, the user must use the scroll [16] because they do not fit in the interface. Therefore, when the service number in a view context exceeds seven, usability decreases.
  - Seven or less navigations from the same context provide plenty of information [18]. Therefore, contexts with more than seven navigations decrease usability because they provide information in excess and not useful.
  - Definition of Filters and Indexes help readability, but they should not contain more than three variables [16] to avoid confusing the user. If the analyst has to provide different alternatives for filtering or indexing, he/she should define several Filters or Indexes that only contains three variables at most.
  - Pagination is another mechanism to make the list of instances readable. Information should be arranged so that the user does not need to use the scroll [16]. Twenty instances fit in an interface at most, therefore more than twenty instances by page decreases usability.
5. Message concision: This attribute indicates whether a message shown to the user is clear. This attribute, which belongs to the understandability sub-characteristic, is significantly related to semantic aspects, which cannot be measured in the Conceptual Model. However, a criterion to automatically measure this attribute could be the number of words used in the message. Error messages should not contain more than 20 words, according to [16].
6. Navigability: This attribute indicates whether navigation is easy and it belongs to the understandability sub-characteristic. Two measures are used for this attribute:
  - The maximum number of navigations to reach a context. According to [16], more than three navigations decrease usability.
  - System functionality should be organized into submenus so that no more than seven menus [18] are shown to the user.

Table 1 shows the metrics defined for each attribute in a formal notation. The attributes are grouped by the sub-characteristics to which they belong.

These metrics provide a number to measure the usability, but this number needs a meaning. Therefore, we must define **indicators** in order to establish how useful the measure obtained by means of the metrics is. For this purpose, we have defined five ranks for each metric. Each one of these ranks represents a qualitative value to show the degree of learnability or understandability of the measured attribute. Depending

**Table 1.** Summary of metrics for each attribute. Notation used: (S)Service; (F)Filter; (C)Context; (A)Attribute.

Sub-characteristic	Attribute	Measure
Learnability	Action determination (AD)	$\forall x \in Alias(S, F, C): \frac{\sum_{i=1}^n x_i}{\sum S + \sum F + \sum C} = AD$
	Labelling significance (LS)	$\forall x \in Alias(A): \frac{\sum_{i=1}^n x_i}{\sum A} = LS$
Understandability	Brevity (BR)	$\forall x, y \in Context : MinPath[x, y] = BR$
	Information density (ID)	$\forall x \in ContextAttribute : \sum_{i=1}^n x_i = ID1$
		$\forall x \in ContextService : \sum_{i=1}^n x_i = ID2$
		$\forall x \in Context : \sum OutDegree(x) = ID3$
		$\forall x \in Context(FilterVariable \vee Index) : \sum_{i=1}^n x_i = ID4$
		$Pagination\_Cardinality = ID5$
	Message concision (MC)	$\forall x \in PreconditionText(Word) : \sum_{i=1}^n x_i = MC$
	Navigability (NV)	$\forall x, y \in Context : MaxPath[x, y] = NV1$
		$\forall x \in Context : Path[Root, X] = 1 \rightarrow \sum_{i=1}^n x_i = NV2$

**Table 2.** Qualitative values for each obtained measured

Measure	VG	G	M	B	VB
AD	$AD \geq .95$	$.95 > AD \geq .85$	$.85 > AD \geq .75$	$.75 > AD \geq .65$	$AD < .65$
LS	$LS \geq .95$	$.95 > LS \geq .85$	$.85 > LS \geq .75$	$.75 > LS \geq .65$	$LS < .65$
BR	$BR \leq 2$	$2 < BR \leq 4$	$4 < BR \leq 5$	$5 < BR \leq 6$	$BR > 6$
ID1	$ID1 \leq 8$	$8 < ID1 \leq 10$	$10 < ID1 \leq 12$	$12 < ID1 \leq 16$	$ID1 > 16$
ID2	$ID2 \leq 5$	$5 < ID2 \leq 7$	$7 < ID2 \leq 10$	$10 < ID2 \leq 13$	$ID2 > 13$
ID3	$ID3 \leq 5$	$5 < ID3 \leq 7$	$7 < ID3 \leq 10$	$10 < ID3 \leq 13$	$ID3 > 13$
ID4	$ID4 \leq 2$	$2 < ID4 \leq 4$	$4 < ID4 \leq 6$	$6 < ID4 \leq 8$	$ID4 > 8$
ID5	$ID5 \leq 20$	$20 < ID5 \leq 30$	$30 < ID5 \leq 40$	$40 < ID5 \leq 50$	$ID5 > 50$
MC	$MC \leq 15$	$15 < MC \leq 20$	$20 < MC \leq 25$	$25 < MC \leq 30$	$MC > 30$
NV1	$NV1 \leq 3$	$4 < NV1 \leq 6$	$6 < NV1 \leq 8$	$8 < NV1 \leq 9$	$NV1 > 9$
NV2	$NV2 \geq .95$	$.95 > NV2 \geq .85$	$.85 > NV2 \geq .75$	$.75 > NV2 \geq .65$	$NV2 < .65$

on the rank in which the numeric value of the metric is, a qualitative value is assigned to the attribute. The qualitative values used are: Very Good (VG), Good (G), Medium (M), Bad (B), and Very Bad (VB). The determination of the ranks was based on certain usability criteria, such as [15],[16],[18] Table 2 shows the indicators for each attribute with a defined metric:

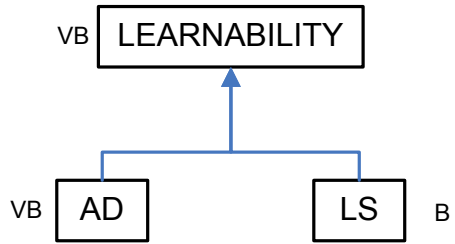
The final step is to establish the usability of the sub-characteristics from the indicators defined for the attributes. Since a sub-characteristic is composed of several attributes, a method to group all the attributes is needed. The method chosen for this is rewriting terms, a method used in Quispe et al work [24]. This method builds a tree with the usability model: the leaves are the attributes, and the branches are the sub-characteristics. Transformations used in the rewriting terms collect leaves two by two to obtain a qualitative value applying a set of rules. The application of these rules is carried out down to top, until the root of the tree, which corresponds to the usability of the system. These rules are the following:

- If we have the Combination of two parameters, one with the Medium (M) value and the other with an X value, the result of this combination is X.

$$\forall x \in \text{Indicators} : \text{Combination}(x, M) \rightarrow x$$

- The rest of combinations are in Figure 3a:

$\text{Combination}(\text{VG}, \text{VB}) \rightarrow \text{M}$   
 $\text{Combination}(\text{G}, \text{B}) \rightarrow \text{M}$   
 $\text{Combination}(\text{VB}, \text{VB}) \rightarrow \text{VB}$   
 $\text{Combination}(\text{VB}, \text{B}) \rightarrow \text{VB}$   
 $\text{Combination}(\text{VB}, \text{G}) \rightarrow \text{VB}$   
 $\text{Combination}(\text{B}, \text{B}) \rightarrow \text{VB}$   
 $\text{Combination}(\text{G}, \text{G}) \rightarrow \text{VG}$   
 $\text{Combination}(\text{VG}, \text{B}) \rightarrow \text{VG}$   
 $\text{Combination}(\text{VG}, \text{G}) \rightarrow \text{VG}$   
 $\text{Combination}(\text{VG}, \text{VG}) \rightarrow \text{VG}$



**Fig. 3.** (a) Combination rules for qualitative values. (b) Example of grouping attributes

For instance, if the measure AD has the indicator VB and the measure LS has the indicator B, the result of the sub-characteristic learnability is VB following the rewriting terms (Figure 3b). The value extracted using the transformation rules for each sub-characteristic comprises the results of the analysis.

All this evaluation method can be carried out automatically. Once the analyst has built the Conceptual Model, he/she can obtain the results of the usability evaluation and change the models in order to improve the usability of the represented system.

## 5 A Priori Validation of the Early Usability Model

According to Morris and Desharnais [19], a priori validation includes two types of verification related to reliability and coherence of the documentation and the training process in order to assure the understandability of the method to be evaluated. In this section, we present the verification of the reliability of the instruments to be used in the evaluation process of the early usability model. Figure 4 summarizes this process, which consists of a comparison between two sets of values: 1) a first set obtained by an analyst applying an early Usability Model within OOWS conceptual schemas, and, 2) a set of values of perceived learnability and understandability for end users using web applications generated from previously evaluated conceptual schemas..

For the gathering of these values from analyst and end users, we prepare various experimental instruments, whose reliability are verified and described below.

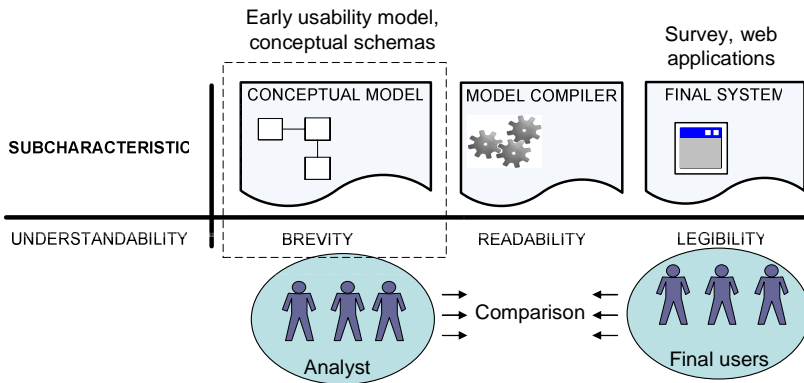


Fig. 4. General process for verifying an early usability model

### 5.1 Verifying the Reliability of the Instruments

With the aim of verifying the clarity of the metrics definition proposed in the early usability model; we selected five senior analysts with extensive experience in the use of the OOWS approach. These analysts measured the conceptual schemas of two case studies. The code from both examples (except aesthetic properties such as colour, font-family, background etc.) was automatically generated by OlivaNOVA and the OOWS Model Compiler:

- *Rent Car*: This web application has been generated from a conceptual model that represents an on-line car rental service. Its application domain is related to e-commerce, as a consequence the rental service must be attractive to potential customers. To simplify the interaction, users can only make car rents and introduce new vehicles into the system. This web application is in <http://oomethod.dsic.upv.es/rentcar>
- *IMDB Lite*: This case study is based on an online repository of information related to the movie world: The Internet Movie Database (IMDB). This case study is a good example of massive information retrieval web. From usability



point of view, is interesting to provide accurate mechanisms to find the required information. Only some functionality from the Website, have been implemented. For instance to see information about movies and actors, etc. and to make comments about them. This web application is in <http://oomethod.dsic.upv.es/imdblite>

Analysing the set of metrics (Table 1), we note a low reproducibility for the NV2, MC, and LS metrics. Other metrics showed a high reproducibility.

With respect to the survey instrument, it was based on existing surveys. These surveys are typically used to obtain information about user perceptions related to achieved satisfaction using prototypes or final software systems, such as WAMMI (Web site Analysis and MeasureMent Inventory) [13], ISOMETRICS [10], SUMI (Software Usability Measuring Inventory) [14], and QUIS (Questionnaire for User Satisfaction)[4]. We adapted these surveys in order to focus solely on the attributes that can be measured within the Conceptual Model (learnability and understandability sub-characteristics).

Our survey instrument included thirteen closed questions: four questions relating to learnability (I1,I5,I7,I11), and nine questions relating to understandability (I2,I3,I4,I6,I8,I9,I10,I12,I13). The items used were formulated using a 5-point Likert scale with the opposing-statement question format. The order of the items was randomized and half the questions negated to avoid monotonous responses. Figure 5 shows two items of the survey.

I3	It is easy to see at a glance the information and options available in a page	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Typically, scroll bars are needed to be able to visualize all the information of a page
I4	Error messages clearly explain the causes of the problems	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	<input type="radio"/>	Error messages do not clearly explain the causes of the problems

**Fig. 5.** Two items of the survey

With the aim of verifying the reliability of the survey, eight users were first requested to accomplish several tasks using two generated Web Applications (*Rent Car* and *IMDB Lite*). *Rent Car* Web Applications had some usability problems while *IMDB Lite* was theoretically usable. Then, in order to capture the users' perceptions using these applications, users completed the specially designed survey.

We used an inter-item correlation analysis to evaluate the construct validity of the learnability and understandability variables. Tables 3 and 4 show the analysis made with data from the surveys of *Rent Car* and *IMDB Lite* applications, respectively. We employed two criteria, Convergent Validity (CV) (first yellow column) and Discriminant Validity (DV) (second yellow column), for each item; if convergent validity was higher than discriminant validity, the item would be validated (last column).

Table 3. Inter-item correlation analysis – *Rent Car* application

		AD			LS		BR		ID		MC			NV					
		I1	I7	I11	I5	I11	I2	I8	I3	I9	I4	I10	I13	I6	I8	I12	CV	DV	VALID
AD	I1	1,00	0,36	0,50	0,52	0,50	0,50	0,22	0,21	0,12	0,13	0,09	0,19	0,45	0,22	0,70	0,62	0,32	Yes
	I7	0,36	1,00	0,69	0,04	0,69	0,03	0,95	0,81	0,15	0,10	0,25	0,18	0,23	0,95	0,73	0,68	0,43	Yes
	I11	0,50	0,69	1,00	0,46	1,00	0,68	0,65	0,68	0,63	0,35	0,40	0,66	0,69	0,65	0,63	0,73	0,62	Yes
LS	I5	0,52	0,04	0,46	1,00	0,46	0,48	0,00	-0,06	0,49	0,70	0,50	0,20	0,17	0,00	0,35	0,73	0,30	Yes
	I11	0,50	0,69	1,00	0,46	1,00	0,68	0,65	0,68	0,63	0,35	0,40	0,66	0,69	0,65	0,63	0,73	0,63	Yes
BR	I2	0,50	0,03	0,68	0,48	0,68	1,00	0,00	0,35	0,53	0,42	0,39	0,65	0,73	0,00	0,11	0,50	0,43	Yes
	I8	0,22	0,95	0,65	0,00	0,65	0,00	1,00	0,81	0,27	0,12	0,19	0,25	0,19	1,00	0,65	0,50	0,46	Yes
ID	I3	0,21	0,81	0,68	-0,06	0,68	0,35	0,81	1,00	0,12	0,34	0,50	0,25	0,26	0,81	0,30	0,56	0,46	Yes
	I9	0,12	0,15	0,63	0,49	0,63	0,53	0,27	0,12	1,00	0,22	0,07	0,88	0,66	0,27	0,34	0,56	0,41	Yes
MC	I4	0,13	0,10	0,35	0,70	0,35	0,42	0,12	0,34	0,22	1,00	0,88	-0,03	-0,16	0,12	-0,12	0,62	0,21	Yes
	I10	0,09	0,25	0,40	0,50	0,40	0,39	0,19	0,50	0,07	0,88	1,00	-0,02	-0,06	0,19	-0,14	0,62	0,23	Yes
	I13	0,19	0,18	0,66	0,20	0,66	0,65	0,25	0,25	0,88	-0,03	-0,02	1,00	0,91	0,25	0,31	0,32	0,45	No
NV	I6	0,45	0,23	0,69	0,17	0,69	0,73	0,19	0,26	0,66	-0,16	-0,06	0,91	1,00	0,19	0,43	0,54	0,40	Yes
	I8	0,22	0,95	0,65	0,00	0,65	0,00	1,00	0,81	0,27	0,12	0,19	0,25	0,19	1,00	0,65	0,61	0,43	Yes
	I12	0,70	0,73	0,63	0,35	0,63	0,11	0,65	0,30	0,34	-0,12	-0,14	0,31	0,43	0,65	1,00	0,69	0,37	Yes

Table 4. Inter-item correlation analysis – *IMDB Lite* application

		AD			LS		BR		ID		MC			NV			CV	DV	VALID
		I1	I7	I11	I5	I11	I2	I8	I3	I9	I4	I10	I13	I6	I8	I12			
AD	I1	1,00	0,65	0,58	0,63	0,58	0,29	0,34	0,61	-0,44	0,27	0,33	0,48	0,61	0,34	0,85	0,74	0,40	Yes
	I7	0,65	1,00	0,18	0,69	0,18	0,44	0,92	0,18	0,10	0,41	0,50	0,73	0,66	0,92	0,55	0,61	0,52	Yes
	I11	0,58	0,18	1,00	0,31	1,00	0,06	-0,07	0,95	0,05	0,34	0,30	0,28	0,38	-0,07	0,76	0,58	0,36	Yes
LS	I5	0,63	0,69	0,31	1,00	0,31	0,31	0,47	0,32	0,08	0,28	0,42	0,27	0,84	0,47	0,63	0,65	0,44	Yes
	I11	0,58	0,18	1,00	0,31	1,00	0,06	-0,07	0,95	0,05	0,34	0,30	0,28	0,38	-0,07	0,76	0,65	0,36	Yes
BR	I2	0,29	0,44	0,06	0,31	0,06	1,00	0,44	0,06	0,00	0,10	0,29	-0,09	0,26	0,44	0,54	0,72	0,21	Yes
	I8	0,34	0,92	-0,07	0,47	-0,07	0,44	1,00	-0,07	0,30	0,45	0,51	0,72	0,43	1,00	0,27	0,72	0,40	Yes
ID	I3	0,61	0,18	0,95	0,32	0,95	0,06	-0,07	1,00	0,11	0,41	0,43	0,24	0,33	-0,07	0,79	0,55	0,40	Yes
	I9	-0,44	0,10	0,05	0,08	0,05	0,00	0,30	0,11	1,00	0,41	0,46	0,11	0,00	0,30	-0,11	0,55	0,10	Yes
MC	I4	0,27	0,41	0,34	0,28	0,34	0,10	0,45	0,41	0,41	1,00	0,95	0,63	-0,10	0,45	0,30	0,86	0,31	Yes
	I10	0,33	0,50	0,30	0,42	0,30	0,29	0,51	0,43	0,46	0,95	1,00	0,51	0,02	0,51	0,43	0,82	0,37	Yes
	I13	0,48	0,73	0,28	0,27	0,28	-0,09	0,72	0,24	0,11	0,63	0,51	1,00	0,24	0,72	0,24	0,71	0,35	Yes
NV	I6	0,61	0,66	0,38	0,84	0,38	0,26	0,43	0,33	0,00	-0,10	0,02	0,24	1,00	0,43	0,62	0,68	0,34	Yes
	I8	0,34	0,92	-0,07	0,47	-0,07	0,44	1,00	-0,07	0,30	0,45	0,51	0,72	0,43	1,00	0,27	0,57	0,41	Yes
	I12	0,85	0,55	0,76	0,63	0,76	0,54	0,27	0,79	-0,11	0,30	0,43	0,24	0,62	0,27	1,00	0,63	0,50	Yes

Item I8 appears in two rows and columns of the tables because corresponds to a question used to perceived two attributes of the usability model: *brevity* (BR) and *navigability* (NV). In a similar way, I11 corresponds to *action determination* (AD) and *labelling significance* (LS). The remaining items perceive just one attribute each one.

In the *Rent Car* analysis (see Table 3), the CV value was higher than the DV value for twelve items. So, the twelve items were validated. However, as item I13 had a CV value lower than the corresponding DV value, this item was not validated.

In the *IMDB Lite* analysis (see Table 4), the CV value was higher than the DV value for the thirteen items, all were validated, including item I13.

In addition, we also conducted a reliability analysis on the items to calculate the degree to which the values of the constructs are free of measurement error. The reliability analysis was carried out using the Chronbach alpha technique; the corresponding alpha value for learnability and understandability sub-characteristics is shown in Table 5.

**Table 5.** Reliability analysis for survey items

Sub-characteristic	Cronbach alpha	Number of Items
Learnability	0.732	4
Understandability	0.804	9
General	0.873	13

These values indicate that the items included in the survey are reliable, alphas of 0.7 or above being acceptable according to Nunally [21].

## 6 Conclusions and Future Work

This paper presents a usability evaluation method applicable to the analysis phase of the OOWS development process. This method is of particular interest within an industrial context because it allows easy improvement of systems usability. After a rapid analysis of usability, the analyst can change the Conceptual Models to improve the usability value. To carry out this process, a set of attributes that can be measured in the Conceptual Model are identified on the basis of a generic Usability Model [7][1], including also attributes that are measured in the Model Compiler or in the final system.

From all the set of attributes, this work is centred in the attributes that can be measured in a Conceptual Model. We define a set of metrics for each one of the usability attributes that can be measured in an early phase (learnability and understandability). A qualitative value is then assigned to each numeric value provided by these metrics in order to provide a meaning to them. This qualitative value is expressed on an ordinal scale with five levels; intervals for each level are defined according to usability criteria revised in the literature [15],[16],[18].

All the qualitative values obtained for each attribute are then grouped in order to obtain the degree of learnability and understandability. This value aggregation is carried out applying a set of rules proposed by Quispe et al.[24].

It is important to note that early usability forms only a part of the measurable usability in that there are many subjective attributes that can be measured only in the final system or in the model compiler. Nevertheless, this early usability evaluation is useful because it helps the analyst to predict the usability of applications generated with OOWS.

Finally, in this paper, we have verified the instruments to be used to evaluate an early usability model. We have found that the items included in the designed survey

are reliable. However, we note that three metrics needed to be revised due to the low reproducibility obtained.

As a next step we plan to carry out an experimental study with Computer Science students in order to ensure that the metrics, indicators and aggregation rules have been correctly defined in order to evaluate the usability of OOWS conceptual schemas.

## Acknowledgments

We would like to thank Alain Abran and the anonymous reviewers for his helpful comments and suggestions that contributed to the improvement of this paper.

## References

- [1] Abrahao, S., Insfrán, E.: Early Usability Evaluation in Model Driven Architecture Environments. In: Sixth International Conference on Quality Software (QSIC 2006), pp. 287–294 (2006)
- [2] Bastien, J.M., Scapin, D.: Ergonomic Criteria for the Evaluation of Human-Computer Interfaces. Rapport technique de l'INRIA: 79 (1993)
- [3] CARE Technologies S.A, <http://www.care-t.com>
- [4] Chin, J.P., Diehl, V.A., Norman, K.: Development of an instrument measuring user satisfaction of the human-computer interface. In: Proc. ACM CHI 1988 (Washington, DC), pp. 213–218 (1987)
- [5] Dix, A., Abowd, G., Beale, R., Finlay, J.: Human-Computer Interaction. Prentice Hall Europe (1998)
- [6] Dromey, R.G.: Software Product Quality: theory, Model and Practice, Technical report, Software Quality Institute, Griffith University, Nathan, Brisbane, Australia (1998)
- [7] España, S., Pederiva, I., Panach, I., Abrahão, S., Pastor, O.: Evaluación de la Usabilidad en un Entorno de Arquitecturas Orientadas a Modelos. In: IDEAS 2006, Argentina, pp. 331–344 (2006)
- [8] Fitzpatrick, R., Higgins, C.: Usable Software and Its Attributes: A Synthesis of Software Quality, European Community Law and Human-Computer Interaction. In: Proceedings of HCI on People and Computers XIII, pp. 3–21. Springer, London (1998)
- [9] Fons, J., P.V., Albert, M., Pastor, O.: Development of Web Applications from Web Enhanced Conceptual Schemas. In: Song, I.-Y., Liddle, S.W., Ling, T.-W., Scheuermann, P. (eds.) ER 2003. LNCS, vol. 2813. Springer, Heidelberg (2003)
- [10] Gediga, G., Hamborg, K.-C., Duntsch, I.: The IsoMetrics Usability Inventory: An Operationalisation of ISO 9241-10. Behaviour and Information Technology 18, 151–164 (1999)
- [11] ISO 9241-11, Ergonomic requirements for office work with visual display terminals - Part 11: Guidance on Usability (1998)
- [12] ISO/IEC 9126-1, Software engineering - Product quality - 1: Quality model (2001)
- [13] Kirakowski, J., Claridge, N.: Human centered measures of success in web site design. In: Proceedings of the Fourth Conference on Human Factors & the Web, Basking Ridge (June 1998)
- [14] Kirakowski, J., Corbett, M.: SUMI: The software usability measurement inventory. British Journal of Educational Technology 24(3), 210–212 (1993)
- [15] Lacob, M.-E.: Readability and Usability Guidelines (2003), <https://doc.telin.nl/dscgi/ds.py/Get/File-35439>

- [16] Leavitt, M., Shneiderman, B.: Research-Based Web Design & Usability Guidelines, U.S. Government Printing Office (2006), <http://www.usability.gov/pdfs/guidelines.html>
- [17] MDA Last visited: May 2007, <http://www.omg.org/mda>
- [18] Miller, G.A.: The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information. *The Psychological Review* 63, 81–97 (1956)
- [19] Morris, P., Desharnais, J.M.: Function Point Analysis. Validating the Results. In: IFPUG Spring Conference, Atlanta, USA (1996)
- [20] Nielsen, J.: Usability Engineering. Morgan Kaufmann Publishers Inc. (1993)
- [21] Nunally, J.: Psychometric Theory, 2nd edn. McGraw-Hill, New York (1978)
- [22] Olsina, L.: Quantitative Methodology for Valuation and Comparison of Web Site Quality. Argentina, Phd. Facultad de Ciencias Exactas Universidad Nacional de La Plata (1999)
- [23] Pastor, Ó., Molina, J.C.: Model Driven Architecture in Practice: A Software Production Environment based on Conceptual Modeling. Springer, Heidelberg (2007)
- [24] Quispe Cruz, M., Condori Fernández, N.: Requisitos No Funcionales: Evaluando el Impacto de Decisiones. In: VI Jornada Iberoamericana de Ingeniería del Software e Ingeniería del Conocimiento (JIISIC 2007), Universidad Pontificia Católica del Perú, Lima (Peru) (2007)
- [25] Shneiderman, B.: Designing the User Interface: Strategies for Effective Human-Computer Interaction. Addison-Wesley, Reading (1998)
- [26] Valverde, F., Valderas, P., et al.: A MDA-Based Environment for Web Applications Development: From Conceptual Models to Code. In: 6th International Workshop on Web-Oriented Software Technologies (IWWOST), Como (Italy) (2007)
- [27] Welie, M.v., Veer, G.C.v.d., Eliëns, A.: Breaking Down Usability. In: INTERACT 1999, pp. 613–620. IOS Press, Edinburgh (1999)

# An Empirical Study of Process and Product Metrics Based on In-process Measurements of a Standardized Requirements Definition Phase

Yoshiki Mitani<sup>1,2</sup>, Tomoko Matsumura<sup>2</sup>, Mike Barker<sup>2</sup>, Seishiro Tsuruho<sup>1,3</sup>,  
Katsuro Inoue<sup>4</sup>, and Ken-Ichi Matsumoto<sup>2</sup>

<sup>1</sup> Information Technology Promotion Agency, Japan (IPA)

<sup>2</sup> Nara Institute of Science and Technology (NAIST)

<sup>3</sup> Kochi University of Technology

<sup>4</sup> Osaka University

{ymitani,tomoko-m}@empirical.jp, mbarker@MIT.EDU,  
tsuruho@ipa.go.jp, inoue@ist.osaka-u.ac.jp, matumoto@is.naist.jp

**Abstract.** This paper focuses on in-process project measurement in the requirements definition phase based on progress with standardization of this phase. The authors have verified the utility of in-process project measurement in a real mid-scale multi-vendor distributed project. This trial was successful, but limited to a part of the total development process. The project measurement target was limited to later processes such as the coding and testing phases where the output products were easy to acquire. The requirements definition phase where process and product were not standardized was difficult to measure. However, a newly provided governmental process guideline standardizes the process and product for the requirements definition phase, and the authors had an opportunity to measure such a requirements definition effort. This paper presents an empirical study of in-process project measurement in the standardized requirements definition phase, verifies the usefulness of this measurement for project management, and reveals the possibility of creating a new software metrics field using these measurements.

**Keywords:** Empirical software engineering, Software process measurement, In-process measurement, Enterprise Architecture, Requirements definition phase measurement.

## 1 Introduction

This paper starts by describing the authors' previous project measurement efforts that target downstream processes following the design phase. Then it explains how the Japanese government guidelines using Enterprise Architecture (EA) methods standardize the process and products in the requirement definition phase, making measurement of this phase possible. Finally, the paper outlines the target project using the EA method. It provides descriptions of the process and product measurement methods used in the project, the measurement results, and observations about the usefulness of these measurements and possible extensions of software metrics based on this study.

## 2 Motivation of This Study

This section outlines the previous in-process measurement work as background for the authors' current study. The authors have previously verified the usefulness of in-process project measurement and feedback to project management in development [1][2]. In the software development process, the authors had focused their interest on design, coding, and testing phases and had separated the measurement targets into two categories such as basic targets and extended targets.

The basic measurement targets are the amount of description products for each phase. The basic analysis targets in the coding phase are the number of transitions of description products. Concretely, these are the numbers of additions, eliminations, and modifications of the descriptions. In the testing phase the numbers of detected faults and the numbers of corrections, along with the numbers of transition are measured.

The extended measurements in the design phases measure the amount of design and design review materials and analyze the transitions in that amount. In the coding phase, the measurement targets are source code entities and their modification histories. In the testing phase, various fault reports are measured. Also, fault cause factors and fault related phases are analyzed.

This empirical study combined the 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.

## 3 Characteristics of the "Requirements Definition" Phase

### 3.1 The Requirements Definition Phase and EA Method

Empirical study and measurements of 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 [3] 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. Section 4 describes such an empirical study.

### 3.2 The Japanese Government's EA Guideline

In 2003, the Japanese government provided an EA guideline for the requirements definition phase to reconstruct and optimize government business and information systems [4]. This guideline provides basic architecture, reference models, and EA products in a four-layer structure of Business Architecture, Data Architecture, Application Architecture, and Technology Architecture for the AsIs and ToBe phases. It provides three phases, AsIs, Optimize, and ToBe, as the project process architecture. It recommends that the management approach use Earned Value Management (EVM) and the Work Breakdown Structure (WBS) method. 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). These diagrams are illustrated in Fig. 1-1 to Fig. 1-4.

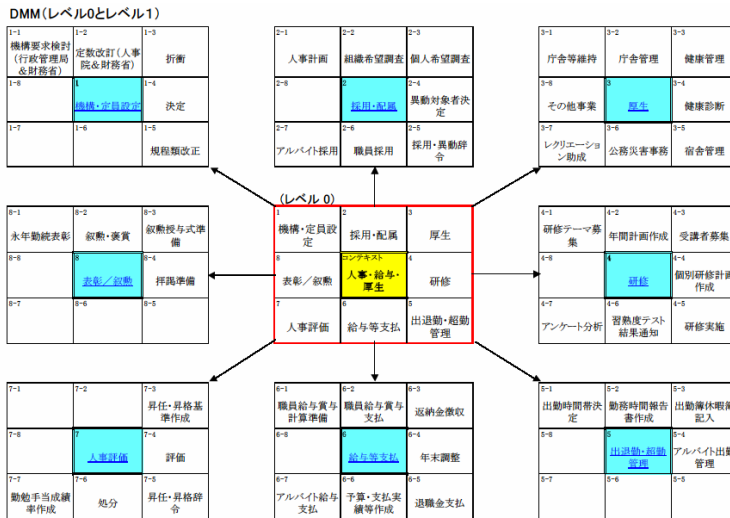
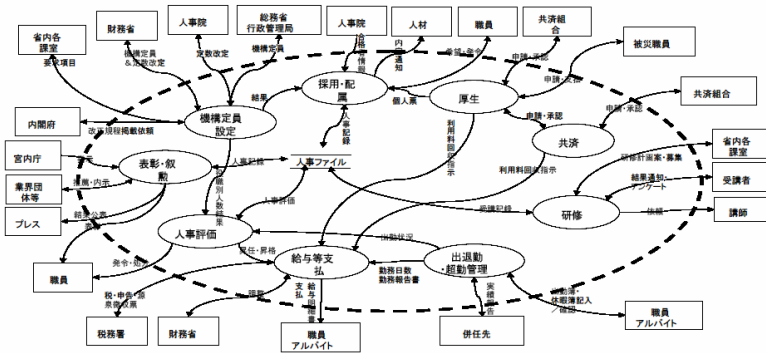


Fig. 1-1. Diamond Mandala Matrix (DMM)

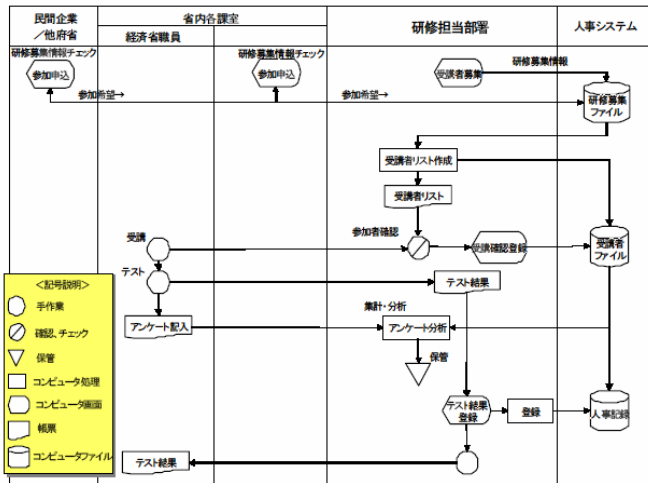
The authors had an opportunity to measure the requirements definition phase of a middle-scale governmental project based on EA methods. This project started in August of 2006 and finished its requirements definition phase in March of 2007.

The next section provides a report and evaluation of the empirical study.

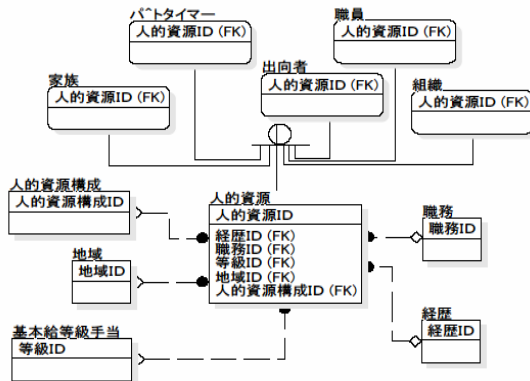




**Fig. 1-2. Data Flow Diagram (DFD)**



**Fig. 1-3. Work Flow Architecture (WFA)**



**Fig. 1-4.** Entity Relationship Diagram (ERD)

## 4 In-process Measurements of an Enterprise Architecture Project

### 4.1 Outline of the Target Project

The target project is a plan for business and system optimization of the Information Technology Promotion Agency, Japan (IPA) [5], a subsidiary organization of the Ministry of Economy, Trade and Industry (METI). This project includes four systems (Businesses A to D) such as a total business management system for a business organization of about 200 employees. IPA staff and a business consulting company accomplished the requirements definition work in a collaborative manner. The goal of the project was to provide requirements definition and an RFP for development process procurement. By government policy, the development phase should be procured from another software company.

This project produced 34 kinds of documents including 29 diagrams as shown in Table 1. The planned document types were different for each business application because the development phase schedule depends on each business.

### 4.2 Project Measurements

This project used the following in-process project measurements:

- 1) Weekly measurement of four types of diagrams: DMM, DFD, WFA, and ERD (18 diagrams total, shaded part of Table 1).
- 2) Interviews of the project leader with a checklist to obtain project context information, a method that was useful in the previous development process measurement study. Also, an interview to the leader after the project finished.
- 3) Attending the project meetings to get context information, also useful in the development process study.

**Table 1.** Outcome Diagrams & Management Targets

Target Business	A		B		C		D	
AsIs/ToBe	AsIs	ToBe	AsIs	ToBe	AsIs	ToBe	AsIs	ToBe
Business Description	x		x		x		x	
<b>DMM</b>			x	x			x	
<b>DFD</b>	x	x	x	x	x	x	x	
<b>WFA</b>	x	x	x	x			x	
<b>ERD</b>			x	x			x	
Information System Reference Diagram			x	x			x	
Network System Diagram			x	x				
Software System Diagram			x					
Hardware Structure Diagram			x					

The measured diagrams were written with an EA tool constructed on Microsoft's drawing tool Visio. The Visio reporting function was used to measure the number of diagrams, including the number of sheets, diagram elements, and transitions. While the measurement process included some manual work, it was mainly done automatically. The 18 documents were selected for study because there were very little other content and few other documents, suggesting that these 18 documents represent the total process outcome for this project. Also the reflection of project context information from 2) and 3) activity is considered very important.

### 4.3 Requirements Definition Phase Measurement Results

In 13 weeks, the description of the AsIs diagrams was completed for three businesses and other one was completed in the end of May 2007. ToBe diagram description for three businesses was started January 2007 and finished by 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. 2 to Fig. 8 shows examples of these measurement results in graphical form. The following trends are directly read from those graphs.

Fig. 2 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. 3 shows the number of AsIs described elements for business B. This study illustrates the data for each business. Fig. 4 and Fig. 5 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. 6 shows all changes 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. 7 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. 8 shows the total changes on a weekly basis. The amount of work performed can be estimated from that figure.

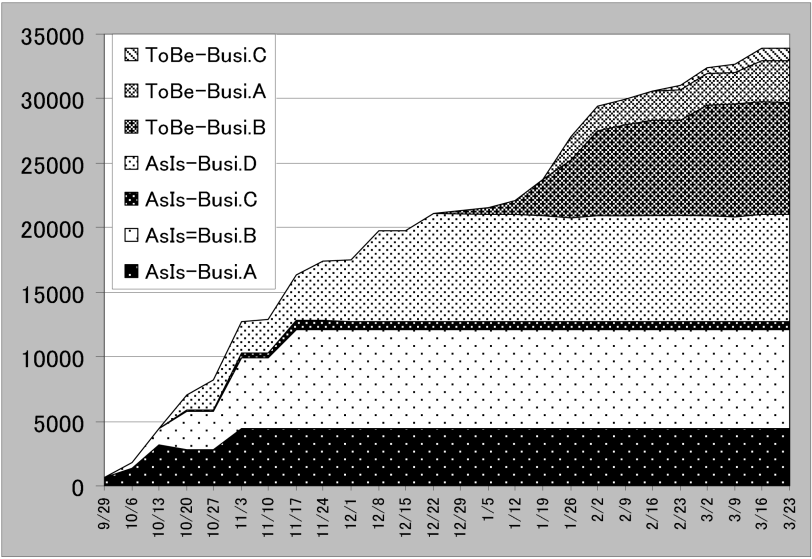


Fig. 2. Diagram Elements Stack of all Business

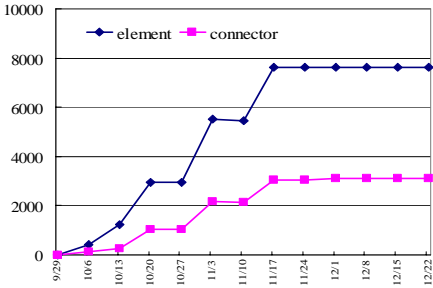


Fig. 3. Diagram Elements of Business B (AsIs)

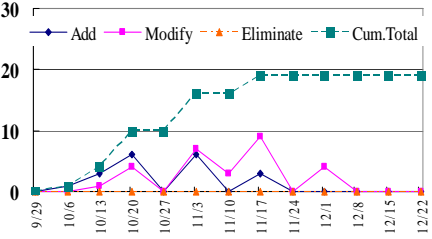


Fig. 4. File Number Transition of Business B (AsIs)

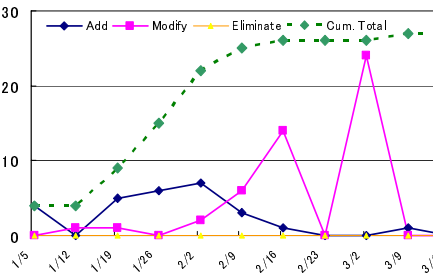


Fig. 5. File Number Transition of Business B (ToBe)

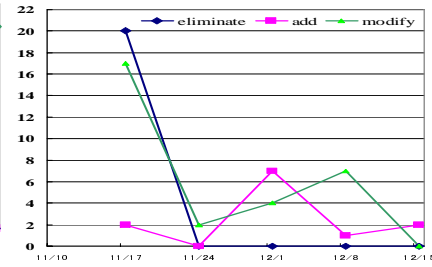


Fig. 6. Diagram Modification in one file Example (8 sheets)

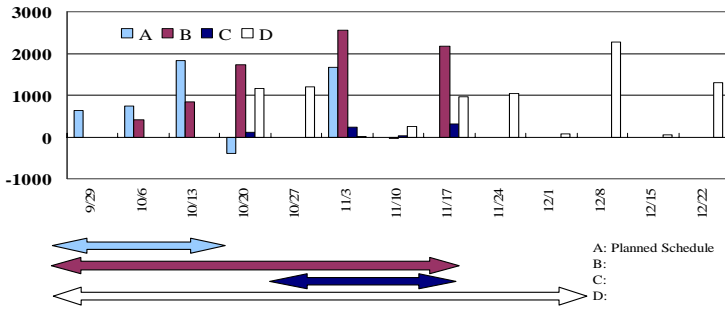


Fig. 7. Weekly Addition of Diagram Elements (AsIs)

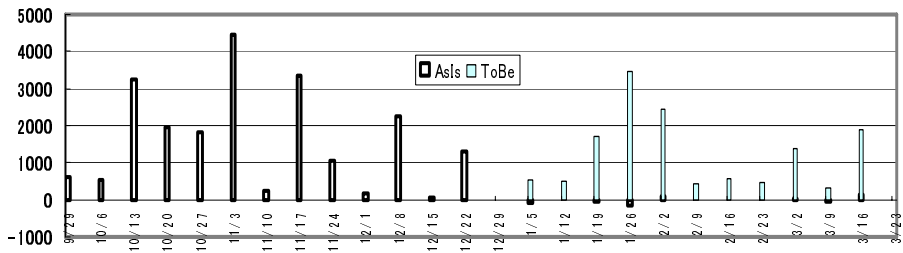


Fig. 8. Total Weekly Addition of Diagram Elements (AsIs and ToBe)

## 5 Evaluation and Study of the Measurement Results

### 5.1 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. 9 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. 10 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. 7 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.

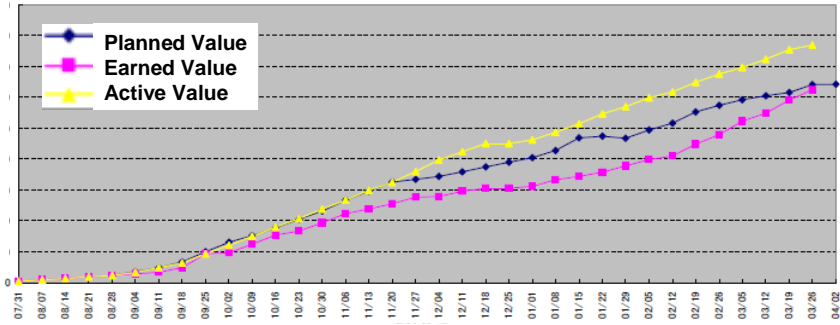


Fig. 9. EVM Report

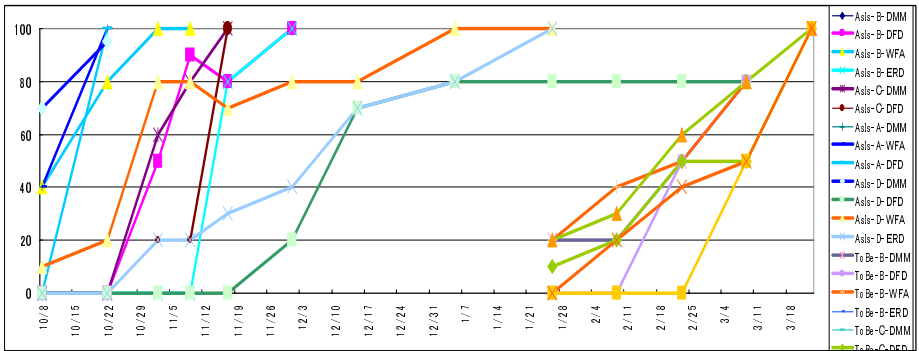


Fig. 10. Declaration Level Progress Report (AsIs & ToBe) (%)

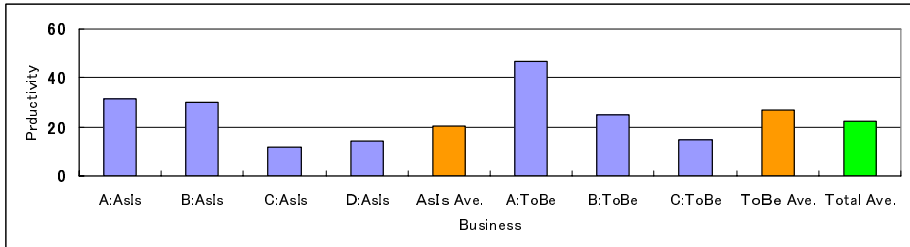
### 5.2 Study for New Software Metrics Possibility

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. 11 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 general trend in the development phase measured by SLOC and Function Point (FP). In the development phase, generally larger development has lower productivity.



**Fig. 11.** Diagram Elements per effort

### 5.3 Possibility as a Software Benchmark Data Element

As previously suggested, from the viewpoint of software metrics the number of diagram elements in the requirements definition phase of the EA method process is strongly analogous to SLOC in the development phase which has meaning for project measurement based on description of the number of transitions.

Various meanings are included in source code and the definition of SLOC is not so strict, but SLOC is often used as an important base for software metrics. For example, in Japan the “Software Development Data White Paper” published every year by the IPA Software Engineering Center includes benchmark data for over 1,000 projects, and shows 213 kinds of analyzed results [6]. In this analysis 75 cases relate to Function Point (FP) and 70 cases relate to SLOC. Comparing SLOC, definition of FP is strict but it is rather static data in software development process and unsuitable to in-process measurement. While SLOC definitions may be imprecise, it is dynamic data in software development and useful for in-process measurement. Similarly, the number of diagram elements will be a useful base for metrics in the requirements definition phase analogous to SLOC in the development phase.

For example Fig. 12 shows cost per element in relative form. (Average is 1). This value varies widely depending on the business. There is difference in AsIs and ToBe. In two businesses, AsIs was higher than ToBe and one business was opposite, and totally ToBe had higher results. This phenomenon was different from the general trend. Generally AsIs phase is easier than ToBe phase, because this work is based on current status of business systems, but ToBe is rather complicated work to design future ideal system. So from this result it is concerned quality of ToBe work. And about ToBe process of business B, it is suggested that this work was done carefully. Also from Fig. 9 and Fig. 10 it appears that the ToBe process was completed rapidly to fit the deadline of project.

This single case study suggests various possibilities for project measurement.

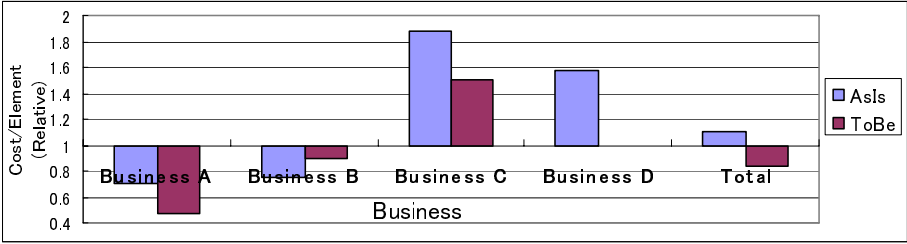


Fig. 12. Cost per Diagram Element (relative value)

### 5.4 Analysis of the Described Process and Diagram Descriptions

As described here, the measurement target did not include all the software processes and products. However, we might hypothesize that the four diagram descriptions do represent the whole process just as SLOC represents the programming effort.

To support this, we can examine the ratio of efforts in the requirements definition phase. As shown in Fig. 13, all of the efforts are defined by 13 work areas. 35% of these are diagram description works of AsIs and ToBe. 23% of them are project management effort, common to the entire work effort. 33% of them are check policy decision and optimize plan making effort, and those two efforts may be considered as included in the ToBe diagram description efforts in wide meaning. So in a broad sense over 90% of efforts are considered as AsIs and ToBe diagram description efforts.

To check this, the ToBe process was further analyzed. Fig.14 shows the comparison of the ToBe diagram description efforts in a broad sense. This compares the direct diagram description effort and the broad sense efforts which include check policy decision and optimize plan making effort and also the common management efforts distributed into all works. Fig.15 illustrates the ratio of both efforts.

Focusing only on the direct diagram description efforts, productivity decreases from business A to B and C in that order. However, including policy decision and planning efforts the order of business A and B is reversed and the productivity of business C is remarkably low. The relationship between business A and B is also clear in Fig.15. Business A has similar characteristics to business C, and that productivity is significantly decreased when it includes policy decision and planning efforts. It shows that the work to clear existing business and plan for future needs certain levels of efforts and those are not in proportion to the narrowly defined direct diagram description efforts.

Focusing on policy decision and planning optimization efforts, process productivity decreases over two to five times and the productivity of the ToBe process is lower than the AsIs process. From this observation three possible cases may be suggested.

Case 1: the policy decision and optimizing planning process were satisfactory

Case 2: the policy decision and optimizing planning process includes a lot of waiting time due to delays in decision making on the procurement side.

Case 3: the policy decision and optimizing planning process consumed a lot of useless effort in no harvest discussion.

This makes clear the limitations of product measurements such as diagram elements in grasping the real situation of the ToBe process. To increase the precision of project metrics, it is necessary to collect more context information such as more observation of project process, project meeting report and review work report and so on.



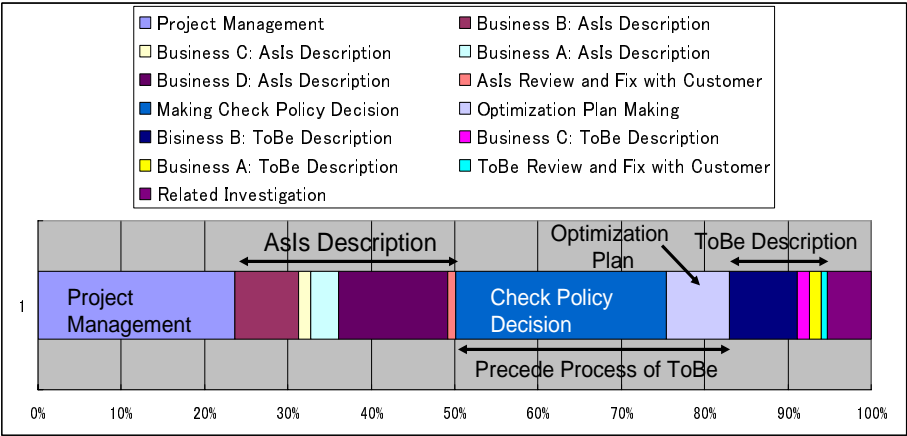


Fig. 13. Working Effort Ratio of Requirement Definition Phase

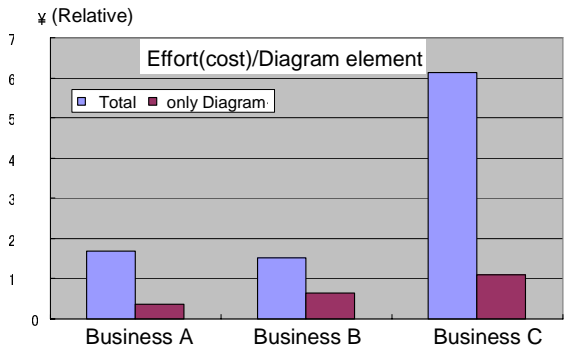


Fig. 14. Comparison of ToBe Total and Only Diagram Description Productivity

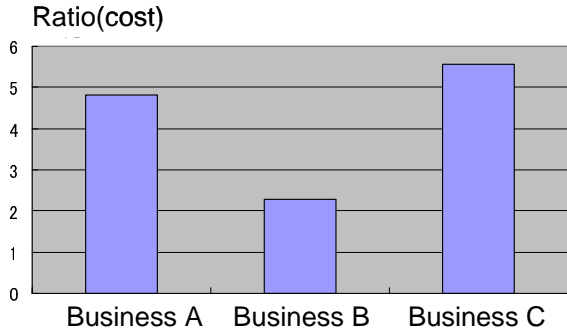


Fig. 15. Ratio of ToBe Total and Only Diagram Description Effort (cost)

## 6 Future Study

This trial had little direct reflection of the measurement results in the project management because this project was planned using traditional management methods with WBS and EVM and these measurements were only additional information. But the measured and analyzed data were useful for managers like the authors who don't have direct access to the real working field. Real evaluation of target project will be clearer at the downstream process.

As in Section 2 shows, these trial measurements remained mostly in basic measurement. Various measurement targets remain in extended measurement. Future studies should look at:

- 1) Collecting quality related data such as review report.
- 2) Collecting user work (effort) data because requirements definitions phase work are collaborations of user and vendor. It is necessary to measure the total effort to evaluate effectively the total project.

## 7 Conclusion

This paper presented an empirical study of process output products measurement in the requirements definition phase where it had been difficult to perform measurements because the process and product were not standardized. The Japanese government standardization of the requirements definition phase using the Enterprise Architecture (EA) method makes it possible to perform various measurements and analyze progress in this phase based on empirical data.

In this trial, the measurement of output diagrams and transitions provided a useful way to characterize project progress. These results suggest that the measured data in the requirements phase can be used in a similar way to SLOC in the development phase, creating a new software metrics field.

**Acknowledgments.** This work is supported by IPA/SEC, METI and the MEXT of Japan, the Comprehensive Development of e-Society Foundation Software program. We thank researchers in the SEC and EASE project who kindly support our project.

## References

1. Mitani, Y., Kikuchi, N., Matsumura, T., Iwamura, S., Barker, M., Matsumoto, K.: An empirical trial of multi-dimensional in-process measurement and feedback on a governmental multi-vendor software project. In: International Symposium on Empirical Software Engineering (ISESE) 2005, Noosa Heads, Australia, vol. 2, pp. 5–8 (November 2005)
2. Mitani, Y., Kikuchi, N., Matsumura, T., Ohsugi, N., Monden, A., Higo, Y., Inoue, K., Barker, M., Matsumoto, K.: A Proposal for Analysis and Prediction for Software Projects using Collaborative Filtering, In-Process Measurements and a Benchmarks Database. In: MENSURA 2006, International Conference on Software Process and Product Measurement, Cadiz, Spain, pp. 98–107 (November 2006)

3. Zachman, J.A.: A Framework for Information Systems Architecture. IBM Systems Journal 26(3) (1987)
4. METI: Enterprise Architecture, <http://www.meti.go.jp/english/information/data/IT-policy/ea.htm>
5. <http://www.ipa.go.jp/index-e.html>
6. IPA/SEC: Software Development Data White Paper 2006 (June 2006) (in Japanese)

# Preliminary Results in a Multi-site Empirical Study on Cross-Organizational ERP Size and Effort Estimation

Maya Daneva

Dept of Computer Science, University of Twente, Enschede, The Netherlands  
m.daneva@utwente.nl

**Abstract.** This paper reports on initial findings in an empirical study carried out with representatives of two ERP vendors, six ERP adopting organizations, four ERP implementation consulting companies, and two ERP research and advisory services firms. Our study's goal was to gain understanding of the state-of-the-practice in size and effort estimation of cross-organizational ERP projects. Based on key size and effort estimation challenges identified in a previously published literature survey, we explored some difficulties, fallacies and pitfalls these organizations face. We focused on collecting empirical evidence from the participating ERP market players to assess specific facts about the state-of-the-art ERP size and effort estimation practices. Our study adopted a qualitative research method based on an asynchronous online focus group.

## 1 Introduction

Requirements specification and architecture design of Enterprise Resource Planning (ERP) solutions mostly takes place in a cross-organizational context [5]. We define cross-organizational ERP systems as information systems that consist of standard ERP software packages and automate cross-organizational process work flows and data control flows, composed of flow fragments owned by or shared among multiple companies. The packages in a cross-organizational ERP system may or may not all be shared by the participating companies; and they may each be provided by the same vendor or by different vendors, each having its own application logic, data formats, and data semantics. Cross-organizational ERP projects tend to have a very high frequency of schedule and cost overruns, quality problems, and outright cancellations [7].

In software engineering, a cost estimation problem is the problem to predict the likely amount of efforts, time, and staffing levels to build a software system [9]. Standard functional size measurement (FSM) models [11,12,13] exist to quantify the functionality that the system provides to its users and to estimate how much effort and time it would take to implement this functionality. Standard FSM models approach the software cost estimation problem by computing a weighted sum of counts of user-recognizable features based primarily on the logical design of the software solution. These counts, then, serve as the key inputs to the prediction of the effort and time needed to build the system [2]. However, the standard FSM models are general solutions initially invented for tailor-made software construction and appear inadequate to the cost estimation problem in ERP implementation settings. Traditional software cost

estimation methods perform poorly in cross-organizational ERP implementation context also because they account for factors which only partially describe this context, and they let partner companies incorporate their bias and intuition into the estimate. Even established approaches such as COCOMO [1] are not suitable to cross-organizational projects [6]. Recent studies [6,14,20] indicate that ERP implementation projects experience a shortage of proper methodologies to evaluate functional size, effort, productivity, schedule, and other cost factors. Even when FSM data exists, effort and duration for similar ERP projects have been noted to vary widely due to (i) variations in the work actually performed on the projects, (ii) variations in the work included in project measurements, and (iii) variations in compensation rates and overhead costs. Each of these three reasons can introduce variances that approach 100% [14]. Yet, without understanding the impact of these three factors, it is not possible to establish productivity averages for cross-organizational ERP implementation teams. In cross-organizational ERP context, these issues are coupled with absence of relevant metrics and historical project datasets. For example, the quantification of reuse of shared data bases and data warehouses as part of an ERP system is a key topic with no metrics at all.

A literature survey [6] we carried out in 2005-06 sketched the problem domain of cross-organizational ERP effort estimation and surveyed existing solutions. Its results were summarized in a list of 10 hypotheses which had implications for both effort estimation analysts and researchers. We used this literature survey as the theoretical foundation for the design of an empirical study which we meant to carry out with four types of organizations who were players in the ERP market.

In this paper, we present some preliminary results from our empirical study. Its overall goal is to explore the state-of-the practice in cross-organizational ERP size and effort estimation. In this study, we focused on the challenges faced by those involved in ERP estimation from the perspective of (i) ERP adopting organizations, (ii) ERP implementation consultants, and (iii) ERP vendors. We got a total of 12 representatives from these three organization types participate in the study. Their perspectives were coupled with insights collected from two representatives from ERP research and advisory firms. We adopted a qualitative research method relying on an online focus group. We, then, grouped the findings into clusters reflecting the structure of our 10 hypotheses from the literature survey [6] to present evidence referring to each hypothesis. However, we make a note, that in this paper, we do not aim at testing the hypotheses formally. Instead, our goal is to build up the “weight of evidence” [19] in support of particular propositions, where evidence is as diverse as possible. As Seaman [19] recommends, we adopt a long-term plan to use the “weight of evidence” to refine the propositions (from our literature survey) to better fit the data. Our motivation for not testing hypotheses is threefold: (a) we have no access to a statistically representative sample of organizations so that we can draw conclusions about the findings in the literature study; (b) we must run some more sophisticated processes for qualitative data analysis [18] in order to derive new knowledge out of the observations that the representatives of the participating organizations provided to us; and (c) building up evidence via qualitative research methods is as instrumental to understanding a software engineering practice as using quantitative methods is [19]. The first two reasons, labeled with (a) and (b), are constraints, which, though detracting from our contribution, do not prevent us from gaining a deeper understanding of

the real-life problems faced by ERP players. At the time of writing, to the best of the author's knowledge, very little research was published in ERP cost estimation and almost no research covered the cross-organizational ERP settings. So, this paper is a first step towards systematically getting insights into how size and effort estimation works (or does not work) in practice. It is a first step of a larger initiative [4] of conceptualizing and organizing the body of knowledge which could be relied on as the foundation for changing the ERP adopters' effort estimation process. In the remainder of this paper, Section 2 summarizes our research method, Section 3 provides clusters of results from our empirical study which address the 10 hypotheses from our earlier literature study [6], and Section 4 discusses some validity threats. Finally, Section 5 reports on the research plan we came up with based on the early results.

## 2 Research Approach

Our approach reflects the notion that the essence of an empirical study is to learn something of value by comparing what we believe to what we see in reality [18]. We set out to investigate the current practice of size and effort estimation in the pre-requirements and the requirements stages of cross-organizational ERP projects. Specifically, we investigated the challenges organizations face from vendor's, ERP implementation consultant's and ERP adopter's perspective, when carrying out business case analysis and estimating functional size and effort as part of it. We adopted a qualitative research approach [8,19] because (i) it best suits our research context, in which there is little previous work and context variables are hard to define and quantify, (ii) it enables an enhanced understanding of why and how aspects of a phenomenon [18], (iii) it fits the setup of explorative studies on 'in-situ' software engineering practices (and cross-organizational ERP cost estimation is of such a nature), and (iv) it is useful in assessing the potential value of future research activities [8], as it gets practitioners involved in the research process. Our qualitative approach implemented (i) an asynchronous online focus group for data collection [10,15,17] and (ii) a member-checking technique [16] for data analysis. A focus group occurs when professionals are brought together to discuss a given topic, which is monitored, facilitated (if needed) and recorded by a researcher. We selected this inquisitive technique because (i) it is known for its cost-effectiveness [15], (ii) it provides ready-to-use transcribed data, (iii) it is flexible so that our group members sitting in various time zones could contribute at their most convenient time, (iv) it encourages candid interchanges and reduces issues of interviewer's effect as focus group members can not "see" each other, and (v) it allows responses that are usually lengthier than in a synchronous mode [17]. The online group comprised 14 members:

- 12 ERP solution architects practicing in Europe and North America and representing two ERP vendors, six ERP adopting organizations, and four ERP implementation consulting companies, and
- two ERP practice analysis representing two US-based ERP research and advisory companies.

All ERP architects were in charge of cross-organizational projects that had stakeholders at locations distributed in at least two organizations. Each architect (i) had at

least 3 years of experience in cross-organizational ERP RE, (ii) has been involved in the estimation of the efforts for cross-organizational projects, and (iii) has done proposals to assess the impact of specific cost drivers on specific projects in the pre-requirements or the requirements stages of their projects. The two ERP practice analysts had observed the trends in the ERP arena since 1997. They were added to the architects (i) because the concept of a focus group implies bringing together people with various backgrounds who are able to see a phenomenon from a variety of perspectives, and (ii) because using multiple data sources ensures data triangulation [19] and helps avoid important validation problems [3].

All focus group members were known to the author, either through common membership in professional societies or through work assignments in which the author was involved with them on a professional basis between 1995 and 2004. They were selected to participate using purposive sampling, based on the author's knowledge and their typicality. They were contacted on a personal basis by the author using e-mail. Before opening the discussion, the author provided the background of this research study and presented the 10 hypotheses as a list of high-level statements put in everyday terms [17]. Focus group data was collected interactively. The type of data being collected was qualitative notes recording our group member's position. The focus group members worked in two stages. First they focused on what they experienced as commonality no matter if they represented an ERP vendor, an adopter, or an implementation partner. Second, they discussed those aspects of the ERP size and effort estimation practices deemed unique to each player. This was to ensure that the group members are not overwhelmed with a long list of inquiries at the start of the process. We also carried out follow-up member-checking activities [16, 19], that is, getting feedback on the findings from the professionals who provided the data in the first place. Their responses are summarized in the next section, in which, for each hypothesis, we present a cluster of observations that offer evidence confirming or disagreeing with this hypothesis.

### 3 Clusters of Results

This section links the observations from our empirical study to the 10 hypotheses from [6].

*Hypothesis 1:* ERP vendors and adopters work together to identify, model, and assess cost factors contributing to the ERP return-on-investment equation. *Observations:* ERP vendors provide to clients' organizations reusable project plans and resource estimates [5]. Published experience reports suggest that these reusable artifacts are created by leveraging large datasets of past projects which ERP vendors carried out at their clients' sites and analyzed per business sector. However, our focus group suggests that the reusable project plans and estimates include, at best, only a partial view of the total ERP costs incurred to ERP adopters. Ten out of 14 members indicate that ERP adopting organizations do not directly provide any project cost data to ERP vendors because of confidentiality concerns. Instead, these members identified the implementation partners of each ERP vendor to be the true suppliers of this data. They also indicated that, what an ERP partner reported to the ERP vendor was the project actuals expressed in "billable consulting staff-hours". These were collected during the

consulting interventions of the consultants employed by the ERP implementation company. Therefore, the focus group admitted this process of project data reporting to reflect solely the perspective of the ERP consulting companies; that is, the project resources employed on the ERP adopter's side were not included. Next, all the six architects from ERP adopting organizations were united on the observation that ERP adopters never relied entirely on the reusable ERP plans provided by ERP vendors. Instead, in their experience, ERP adopters always tried to adjust the reusable plans and estimates based on "what they knew" at the pre-requirements and requirements stages of their project. However, the architects found that adjusting reusable cost estimates took into account obvious cost drivers only, and omitted the ones which were "not so obvious", such as "internal resources required to support the project team, costs to backfill the day-to-day work of project team members, process improvement, hardware upgrades, training, and organizational change". Last, we found that no ERP adopter had any historical database of past projects, accounting for the "not so obvious" cost drivers. None has ever worked with external consultants on formulating a cost estimation model.

One out of the six architects working for ERP adopters suggested that person-hours were recorded according to a corporate-wide data collection procedure for ERP projects. The others witnessed ERP project data treated no differently from other capital (IT and non-IT) project data, which meant that project information was collected regardless of its relevance to the project context.

Two group members out of 14 said that their organizations tailored the ERP project to the resources that they were prepared to commit to the project. When these ERP adopters were unable or unwilling to commit this level of resources, the project was scaled down accordingly and the time scale extended. Three other members witnessed ERP adopters building up their centers for ERP excellence who were given the responsibility for ERP project reporting and tracking. These centers had tool-supported data collection procedures for each ERP project type they handled, namely, new implementations, upgrades, system instance consolidations. They used the datasets of past projects of each type for judging the amount of person-hours needed for future projects of this type. Their projected estimates, however, were never reported to consultants, but did serve as input to cost negotiations with ERP implementation partners. The latter were receptive to the ERP adopter's cost projections and adjusted their numbers. Four out of 14 group members found that ERP adopters, generally, live with whatever estimates they receive from their external consultants.

*Hypothesis 2:* ERP vendors, consultants and adopters are currently applying the cost estimation paradigm in which cost is relative to size of the solution built. *Observations:* All four architects from ERP implementation consulting firms pointed out that for each new project they first looked on how much of an ERP module it would get implemented and then, they defined how many billable staff-hours seemed to be enough for carrying out the implementation tasks. They base their early estimation on their firms' knowledge of (i) the skills of the specific consultant and (ii) his/her past performance in implementing the specific module in which he/she is a specialist. However, the consultant's past experience is rarely collected within one business sector and in organizations of the same size and culture. The fact that in each project, consultant's performance varies due to context factors in the ERP adopting organization (e.g. ERP adopter's culture, business priorities, amount of customization) is, by



and large, ignored in the estimate. Moreover, the focus group was divided on what the term 'size' means. Seven members saw size as the attribute of the tasks it would take to implement a ERP solution. They saw costs as a result of a work-breakdown-structure-based process for estimating the total effort needed for executing these tasks. Five focus group members saw size as the attribute of the user community to be served. They calculated cost by multiplying the number of users by the dollar value per user. For example, SAP, PeopleSoft, and Oracle are known among architects to cost \$80,000 per user, while BaaN and JDE, are estimated for \$40,000 per user. They also added a one-time capital cost of the software which varied enormously - from about US\$10,000 to as much as US\$60,000 per seat/user license, depending on the complexity of the software, the size of the company, and the software and the database license fee. Only two out of 14 focus group members saw size as the attribute of ERP functionality. Both, however, saw functional size as a key driver in cost estimation. Both used some versions of FPA specifically adapted for their package contexts. These two versions, though, took different types of project deliverables (in addition to the requirements) as inputs into the sizing process. The two versions interpreted the standard FSM model [11] differently in terms of what to count and how to count it. In both, FP data were coupled with rules of thumb, for example "a project team of three people is necessary for companies with up to 300 employees" and "one more project team member is needed for every 200 employees".

*Hypothesis 3:* ERP adopters treat the cost for ERP solutions as the cost of doing business; as such, ERP project budgets are approved without much up-front thinking.

*Observations:* The important implication of this proposition is that ERP cost numbers would be accepted no matter how big or precise they are. Two out of 14 focus group members confirmed that ERP project costs were considered as the price of doing business in their markets. The other two saw ERP costs as the price for doing business differently. "All our key competitors have it; we can not afford to lag behind" was a recurrent comment focus group members shared. They attributed this attitude due to the fact that an ERP is as much a strategy as a software system, which assists with the development of information strategies. That is, getting the system in is the cost which the ERP adopter pays for having the potential to develop these strategies. Our group members witnessed too many times, when CIOs who were "so in awe of the whole ERP concept that they want to implement it", no matter how much it costs or how little of a return-on-investment it delivers. Four ERP adopters got involved in what we call the "ERP sales trap", that is, they let their ERP implementation partners convince them that the cost would not be as high as they might think. Five architects also experienced that their project teams just didn't know any better and they overlooked costs.

*Hypothesis 4:* Business case analysis is the most common analysis ERP adopters are doing. *Observations:* Business cases are about assessing cost versus benefits before initiating a project. The focus group members confirmed that "making the business case is the only ritual none can skip". Their stand was that cost estimation discussions on cross-organizational ERP should always be put in context of benefits realization. They argued that (i) ERP cost should be seen in the light of how a shared ERP solution was managed in the long run, and that (ii) how it was managed generated benefits and advantage, not just the fact that a shared ERP system existed. Though,

they remained divided on what cost and benefits the business case analysis should include. In their practice, technology made a company more efficient and this was supposed to ultimately result in an overall headcount reduction. However, they pointed out that there were costs associated with reducing staff, such as severance and business reengineering costs. They indicated that, even though they saw long-term benefits associated with making employees more efficient and effective as a result of the new system, there was a short-term decrease in efficiency as employees were learning on the jobs. Though, an ERP adopter may or may not quantify these cost aspects in their business case.

An unexpected and interesting finding from our focus group discussion was that 11 out of 14 members were involved in business case analysis for “free” ERP systems. They indicated 17 ERP systems available for free to organizations. Despite their costs seemed to be zero, the most important question architects faced was whether it made sense to implement one of these systems. They said, ERP systems were deemed mission-critical to any business and no organization wanted to take any chances with relatively unknown developers or systems that might require a lot of fine-tuning to work for them. Their business cases found that the expensive effort to get these “free” systems up and running made them pricier than the systems which could be purchased/licensed or rented. However, because no architect had any “free ERP” project experience, they were concerned that their cost estimates could be exaggerated.

*Hypothesis 5:* For ERP adopters, business case analysis is a one-time exercise. *Observations:* Ten out of 14 members witnessed business case analysis happening at the project start only. They indicated the business case was used for nothing more than just convincing top management to approve the project. Their organizations justified this with the fact that in alternative ERP solutions, there was very little to distinguish feature-wise.

*Hypothesis 6:* Ongoing business case analysis is more effective than a one-time analysis at the project start. *Observations:* Four out of 14 members kept doing business case analysis at each stage of the project. All four attributed this to the higher level of maturity and “project-cost-and-benefits- consciousness” of their employers. They argued that this was, indeed, an analysis and not a justification, because they used it in a “gating process” to decide to abandon or re-scope the project, should the business case analysis was unfavorable. They also re-visited the analysis after project completion to ensure they cashed all the benefits. Two of the four members used the ongoing business case analysis to identify and manage operational business benefits and key performance indicators during and after the implementation. For example, ongoing analysis clarified costs and benefits associated with getting the accuracy of bill of material and inventory records above the 98%, which was vital to get meaningful data from the ERP system. The analysis revealed that for inventory records, the cost of cycle counting turned out to be on-going. In contrast, once the process for maintaining the bills of material has been established, bills of material remained accurate without any significant on-going cost.

*Hypothesis 7:* Cost is driven by those requirements that are unknown at the stage of requirements. *Observations:* Our findings supported this. The focus group traced most of what they labeled “unknown” back to the amount of business change the project instilled in the adopting organizations. They indicated that cross-organizational ERP

favors ERP instance consolidation projects and this implies much business change. They argued that these ERP adopters initiated their ERP implementations under duress for Y2K reasons and they had no time to optimize business process when it was implemented, and are now ready to get it right now. These focus group members found that achieving cross-organizational business process integration did not require replacement of their ERP but it usually called for adding “last mile functionality or third party software”. As typical examples were given any master data management (MDM) solution or any search tool (such as the Google Search Appliance) for finding information in cross-organizational ERP system(s). In each of these cases, ERP adopters screened different solution options, each one incurring different costs to their projects. Though, ERP adopters indicated that these costs were unknown until later project stages. For example, MDM refers to solving the problem of different systems used to store and maintain master data such as customer or supplier information. This could be a huge problem for larger cross-organizational projects that used multiple ERP systems but it was only in the configuration stage, when the true scope of this problem revealed.

*Hypothesis 8:* ERP sizing is done based on the business requirement document. *Observations:* Our findings confirm that business requirements serve as the key input into cost estimation. However, the experiences of the focus group members varied in terms of what they called “adequate requirements” and how much detail was included in their business requirement documents. ERP vendors’ representatives argued that detailed process and data models should be the first to refer to, when estimating size, and later, effort. These models “were supposed to customize” the work breakdown structure in the standard reusable project plan and estimates provided by the ERP vendor. Consulting companies’ and ERP adopters’ representatives were firm that using the process and data requirements models would render the sizing process inefficient and expensive because estimation analysts at adopters’ sites are not educated on the use of package-specific process modeling and data modeling languages. To set up a meaningful FSM process, these analysts depended on the availability of experts able to read and interpret the process and data diagrams that describe the ERP solution.

*Hypothesis 9:* The diversity of customization options is a key cost driver. *Observations:* 11 out of 14 group members suggest that customization need not be the big problem that it used to be. In their experience, modern systems let one make changes to system’s screens outside of source code so that there is no need to redo them when system upgrades are released. Data reporting should also not be a problem using a report writing tool such as Crystal Reports. The focus group was united on that the configuration staff may add new fields but would rarely change any existing fields. They indicated, there was no reason for reports not to work when upgrades were provided as the tables and fields remained the same. Next, the focus group agreed on the fact that every conscious step should be made to resist the temptation to customize the package, and that there may have to be some changes which should be allowed for. They also shared the view that some packages (e.g. SAP R/3 and JDE’s EnterpriseOne) may be cheaper for the software but much pricier to configure. In their experience, SAP R/3, for instance, may only cost US\$4,500 for the software per user license but US\$20,000 per user license for consultants to configure. An unexpected insight came from 11 out of 14 members who suggested that customization is a “big problem” when an adopter is forced to migrate to

another release or a package. In this case, “ERP adopters are left at the mercy of the vendor’s choice on a version of a system”. They meant it when either an ERP vendor discontinues a piece of functionality in a new release or when a module may be retired due to mergers between ERP vendors. In the latter case, the ERP adopter may well undergo a migration to an ERP suite they purposely did not choose in the first place, which means forcing the adopter “to expend vast amounts of money on customizing an unfamiliar package and on re-training staff, while placing their core business processes at risk”.

*Hypothesis 10:* Knowledge management is a key driver in cross-organizational ERP projects. *Observations:* Ten out of 14 members confirmed that including knowledge transfer as a specific deliverable in the contract with an ERP consulting firm or other external resources could be challenging, but improved the probability of success. In cross-organizational settings, “the rapid syndication of the enterprise process logic, information assets and collaborative subsystems requires mastery in knowledge transfer to be effective and competitive”. The focus group members thought that robust knowledge transfer processes deliver rapid assimilation, improved competitiveness and high performance of all ERP stakeholders and the entire cross-organizational ERP systems portfolio. Therefore, they suggested this be a critical factor in ERP deployment, use and maintenance. Both ERP adopters and consultant’s representative argued that for effective knowledge transfer, external resources must be co-located with internal resources. The two ERP practice analysts witnessed successful ERP efforts set up a “war room”, that is, a large, shared space where all resources work. That enabled a “next bench” sharing practice while ensuring a productive, collaborative work and learning environment. However, this was not found to come cheaply.

Furthermore, the architects raised the concern that ERP projects should explicitly incorporate the costs of tools instrumental to “ERP knowledge diffusion”, like document management systems and other leading collaborative technologies providing full, seamless access to all ERP stakeholders, particularly external consultants. Focus group members varied in terms of what “new” knowledge ERP adopters and consultants required in a dynamic cross-organizational ERP environment, how adopters can better “retain” knowledge during ERP implementation, what respective roles adopters, consultants, and ERP vendors can play in capturing, transferring and managing this knowledge, and what cost levels are associated to this.

## 4 Validation Threats

At this early stage of our study, we could address some validity threats by offering a preliminary assessment only. We did approach two validity issues that can call into doubt the results of our preliminary study or the conclusions from our results. This was done by applying the process described in [21] that suggests researchers focus on two types of validity threats: First, the major threat to external validity arises from the fact that the ERP projects in which the architects were participating, might not be representative for the entire population of cross-organizational ERP projects. Despite the fact that the professionals were chosen based on their typicality (see Section 3), we believe that the size-and-efforts estimation practices, which these architects witnessed in their project settings, may well be just a fraction of what is observable in

real life. We plan a replication study in the Netherlands to bring us to a more exhaustive and a more detailed list of how ERP adopters, vendors and consultants use FSM and effort estimation model and integrate them into larger decision-making processes. Moreover, we are interested in learning from practitioners “how common is common” and in which project context. That is, we would like to gain understanding of those size and effort estimation practices that are specific to new projects, to ERP updates, and to ERP instance consolidation projects.

Second, the key threat to the internal validity is concerned with any “alternative explanations” for the observations on the size and effort estimation practices. That is to say that while analyzing the observations, the practices that all focus group members found to be common, were specific and the final agreement on the commonality was only due to coincidental factors. To make sure we have evidence that what ERP adopters and consultants observed as common was also identifiable by external analysts, we included some representatives of ERP market research companies. This validity aspect is included in our future analysis of the focus group’s transcripts and observations.

## 5 Conclusions and Future Research Plans

This paper presents preliminary clusters of observations used to build up the “weight of evidence” to support propositions from a literature survey. The clusters let us conclude the following about state-of-the-art cross-organizational ERP size and effort estimation practices:

- (i) if any complete information on incurred ERP project costs exists, it is most likely to be found with the ERP adopters;
- (ii) size is defined as an attribute of ERP implementation tasks, business user communities, or solution functionality; the variety of definitions used by our focus group members reflects the current confusion in the FSM literature on the object “being measured” (that is, what to count and how to count it).
- (iii) whenever used, functional size is only one of the many cost parameters considered in ERP business case analysis,
- (iv) the ratio of package-acquisition-cost versus package-customization costs is specific to each ERP package,
- (v) the amount of business change and knowledge management add up to more expensive implementations.

Of course, we see these conclusions as early as our preliminary findings we used to derive them, and therefore, are subjected to further validation studies. As our immediate step, we plan to use sophisticated software tool-supported data analysis techniques (like coding) that will help us trace conclusions to focus group members’ statements.

Second, our preliminary results indicate that the size and effort estimation topic is industry-relevant and, therefore, we also plan to replicate the focus group set up by using subjects representing the Dutch ERP market.

Third, we plan to use this study to catalogue research question that warrant future PhD research project efforts. At the time of writing, the author arrived at a set of 40 research questions with the help of the focus group. These remain to be grouped in areas of focus and, then, prioritized. Ongoing research is being carried out by the

author and members of the COSMOS team [4] to arrive at good [8,18,19] research questions.

**Acknowledgements.** The author thanks the professionals for volunteering their time in the focus group sessions and the anonymous reviewers for suggesting ideas on how to improve this paper. The author also thanks Janice Leschinsky for the inspirational conversations which led to a better design set-up.

## References

- [1] Boehm, B., Abts, C., Chulani, S.: Software Development Cost Estimation Approaches – a Survey. *Annals of Software Engineering* 10, 177–205 (2000)
- [2] Bourque, P., Oligny, S., Abran, A., Fournier, B.: Developing Project Duration Models in Software Engineering. *Journal of Computer Science and Technology* (2006)
- [3] Brathall, L., Jorgensen, M.: Can You Trust a Single Data Source Explorative Software Engineering Case Study. *Journal of Empirical Soft. Eng.* 7, 9–26 (2002)
- [4] Daneva, M.: Status Report on Functional Size Measurement for Cross-organizational ERP Solutions: Problems and Alternative Approaches, IWSM, Potsdam, pp. 423–434 (2006)
- [5] Daneva, M., Wieringa, R.J.: A Requirements Engineering Framework for Cross-Organizational ERP Systems. *Requirements Engineering Journal* 11(3), 194–204 (2006)
- [6] Daneva, M., Wieringa, R.J.: A Conceptual Framework for Research in Cross-organizational ERP Cost Estimation. In: Workshop on Requirements Engineering and Project Management in Software Projects, in conjunction with the 13th IEEE Requirements Engineering Conference (RE 2005), Paris (2005)
- [7] Davenport, T.: *Mission Critical: Realizing the Promise of Enterprise Systems*. HBS Press (2000)
- [8] Dittrich, Y., John, M., Singer, J., Tessem, B.: Editorial for the Special Issue on Qualitative Software Engineering Research. *Journal of Inf. and Soft. Technology* 49(6), 531–539 (2007)
- [9] Fenton, N.E., Pfleeger, S.L.: *Software Metrics, A Rigorous and Practical Approach*, Thomson, 2nd edn. (1996)
- [10] Gaiser, T.: Conducting Online Focus Groups: a Methodological Discussion. *Social Science Computer Review* 15(2), 135–144 (1997)
- [11] Garmus, D., Herron, D.: *Function Point Analysis: Measurement Practices for Successful Software Projects*. Addison Wesley (2001)
- [12] International Standard Organization (ISO), ISO/IEC 19761: Software Engineering – COSMIC-FFP – A functional size measurement method (2003)
- [13] ISO/IEC IS 24570 Software Engineering - NESMA functional size measurement method version 2.1 - Definitions and counting guidelines for the application of Function Point Analysis (2003)
- [14] Jones, C.: *Estimating and Measuring SAP Applications with Function Point Metric*, Software Productivity Research, Burlington (1998)
- [15] Kivits, J.: Online interviewing and the research relationship. In: Hine, C. (ed.) *Virtual methods: issues in social research on the internet* Oxford. Berg, pp. 35–49 (2005)
- [16] Lincoln, Y.S., Guba, E.: *Naturalistic Inquiry*. Sage, Thousand Oaks (1985)
- [17] Orgad, S.: From online to offline and back: Moving from online to offline relationships with research informants. In: Hine, C. (ed.) *Virtual Methods: Issues in Social Research on the Internet*, Oxford, pp. 51–65 (2005)

- [18] Perry, D., Porter, A., Votta, L.: Empirical Studies of Software Engineering: a Roadmap. In: Future of Software Engineering, pp. 345–355. ACM Press (2007)
- [19] Seaman, C.: Qualitative Methods in Empirical Software Engineering. *IEEE Transaction on Soft. Eng.* 25(4), 557–572 (1999)
- [20] Stensrud, E.: Alternative Approaches to Effort Prediction of ERP Projects. *Information & Software Technology* 43(7), 413–423 (2001)
- [21] Yin, R.: Case Study research: Design and Methods. Sage Publications (2003)

# Do Base Functional Component Types Affect the Relationship between Software Functional Size and Effort?

Cigdem Gencel<sup>1</sup> and Luigi Buglione<sup>2</sup>

<sup>1</sup> Bilgi Group Software Research, Training, Consultancy Ltd., Middle East Technical University  
Teknokent, Ankara, Turkey  
cgencel@bg.com.tr

<sup>2</sup> École de Technologie Supérieure (ETS) / Engineering.it S.p.A.,  
luigi.buglione@eng.it

**Abstract.** One of the most debated issues in Software Engineering is effort estimation and one of the main points is about which could be (and how many) the right data from an historical database to use in order to obtain reliable estimates. In many of these studies, software size (measured in either lines of code or functional size units) is the primary input. However, the relationship between effort and the components of functional size (BFC – Base Functional Components) has not yet been fully analyzed. This study explores whether effort estimation models based on BFCs types, rather than those based on a single total value, would improve estimation models. For this empirical study, the project data in the International Software Benchmarking Standards Group (ISBSG) Release 10 dataset, which were functionally sized by the COSMIC FFP method, are used.

**Keywords:** Functional Size Measurement, Effort Estimation, COSMIC-FFP, Base Functional Component, International Software Benchmarking Standards Group (ISBSG).

## 1 Introduction

The planning, monitoring and control of software development projects require that effort and costs be adequately estimated. However, some forty years after the term “software engineering” was coined [24], effort estimation still remains a challenge for practitioners and researchers alike.

There is a large body of literature on software effort estimation models and techniques in which a discussion on the relationship between software size and effort as a primary predictor has been included, such as [2][5][6][11][12][13][14]. Other factors related to non-functional characteristics of software projects – also referred to as cost drivers in the Constructive Cost Model (COCOMO) estimation models and their variants [5][6] – are also included in many estimation models.

In [18], Leung and Fan discuss both the strengths and weaknesses of effort estimation models. They evaluate the performance of existing models, which they deem



unsatisfactory, as well as that of newer approaches to software estimation. Similarly, in a number of studies, such as [2][15][16][17], related work on effort and cost estimation models is assessed and compared. These studies conclude that the models, which are being successfully used by different groups and in different functional domains, have still not gained universal acceptance. And none of the models is considered to perform well enough to fully meet market needs and expectations.

In particular, the nature of the relationship between functional size and effort has been explored in many studies. The common approach of these studies is that the functional size of a software system is expressed as a single value obtained by a specific Functional Size Measurement (FSM) method. This single value is derived from a measurement function in all ISO-certified FSM methods, and it is the result of adding together the functional sizes of different Base Functional Component (BFC)<sup>1</sup> Types to obtain a total functional size.

In the study presented here, effort estimation models based on the functional size of BFC Types rather than the total functional size are explored to investigate whether or not they improve estimation reliability. Our hypothesis is that the effort required to develop the unit size of each of the BFC Types, which provide different user functionalities, is different.

For the statistical analyses, the project data, which were measured by the Common Software Measurement International Consortium's Full Function Points (COSMIC-FFP) [41]<sup>2</sup> and contained in the ISBSG Dataset Release 10, are used [10].

The paper is organized as follows: Section 2 presents some background on functional size measurement and related work on its relationship to project effort. Section 3 presents the data preparation for further statistical analysis. Section 4 presents the data analysis results and section 5, a summary.

## 2 Background

### 2.1 Functional Size Measurement – A Brief Outline

Function Point Analysis (FPA) was designed initially in 1979 [1] by Allan Albrecht, an IBM researcher. 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 was based on the idea of determining size based on the functional requirements and from the end user's viewpoint, taking into account only those elements in the application layer that are logically 'visible' to the user and not the technology used.

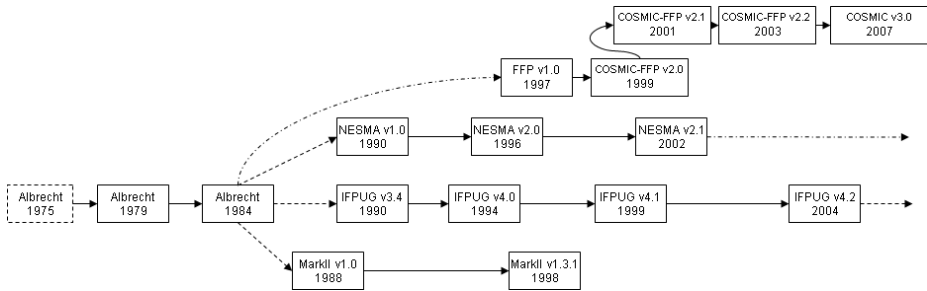
FPA was designed in an MIS environment and has become a *de facto* standard in the MIS community. However, it generated a large number of variants for both MIS and non-MIS environments (such as real-time, Web, Object Oriented, and data warehouse

---

<sup>1</sup> BFC Type: A defined category of BFCs. A BFC is an elementary unit of an FUR defined by and used by an FSM method for measurement purposes [27].

<sup>2</sup> Recently updated to version 3.0 (September 2007). The name of the method has become simply "COSMIC". All information, documentation and case studies are available at [www.cosmicon.com](http://www.cosmicon.com).

systems)<sup>3</sup>. In the '90s, work was initiated at the ISO level to lay the foundations for regulating *de jure* standards in FSM, and the 14143 family was developed [27]<sup>4</sup> [29]–[33] with four instantiations matching with those requirements: COSMIC-FFP [41][34], the International Function Point Users Group (IFPUG) FPA [39][35], MarkII FPA [40][36] and The Netherlands Software Metrics Association (NESMA) FSM [37] methods. The evolution of FSM methods is shown in Fig. 1.



**Fig. 1.** Evolution of the main Functional Size Measurement (FSM) methods

COSMIC-FFP [41], adopted in 2003 as ISO 19761 [34], has been defined as a 2<sup>nd</sup> generation FSM method as a result of a series of innovations, such as: a better fit with both real-time and MIS environments, identification and measurement of multiple software layers, different perspectives (viewpoints) from which the software can be observed and measured, and the absence of a weighting system.

## 2.2 Related Work

### 2.2.1 Functional Profiles and the Size-Effort Relationship

Abran et al. [3] used the 2003 version of the ISBSG repository to build estimation models for projects sized by the FPA method. They defined the concept of a software functional profile as the distribution of function types within the software. 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. For each sample, it was noted that there was one function type that had a stronger relationship with project effort. Moreover, the sets of projects located within a certain range of the average profile led to estimation models similar to those for the average functional profile, whereas projects located outside the range gave different regression models, these being specific to each of the corresponding subsets of projects.

In [4], the impact of the functional profile on project effort was investigated using the ISBSG repository. The ISBSG projects included in this analysis were all sized by the COSMIC-FFP method. For the COSMIC-FFP method [41], a functional profile corresponds to the relative distribution of its four BFC Types for any particular

<sup>3</sup> Please refer to [38] and [8] for a detailed list and a history of FPA-like methods.

<sup>4</sup> Part 1 (14143-1) has recently been updated (February 2007) [28] from its first release [27] (1998).

project. 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 [7], Gencel identified the types of functionalities a software system can provide to its users are identified, and defined a multi-dimensional measure which involves measuring the functional size of each functionality type. Gencel suggested that experimental studies should be conducted to find the relationship between the functional size of each functionality type and the effort needed to develop the type of functionality that can pioneer new effort estimation methods.

### 2.2.2 Project Cost Drivers – the Size-Effort Relationship

In published studies, significant variations in the impact of other project cost drivers have been observed, and therefore a number of experimental studies were performed to investigate their impact on the size-effort relationship. 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 [19][20][21][22][23][25]. Among these, the most significant are reported in [19][20] to be Team Size, Business Area Type and Application Type.

## 3 Data Preparation

In this study, the project data in the ISBSG 2007 Repository, CD Release 10 [10], are used for statistical analysis. This ISBSG Repository includes a large amount of high-quality data on a very wide range of projects. ISBSG Release 10 contains data from 4,106 projects, 117 of which were sized using COSMIC-FFP. The projects cover a wide range of applications, development techniques and tools, implementation languages, and platforms.

Table 1 shows the filtration process we performed on the data which takes into account project data quality attributes defined in the ISBSG dataset.

The first step was to filter the dataset with respect to the ‘Count Approach’ attribute in the ISBSG repository to obtain only the projects measured by COSMIC-FFP. This step provided 117 projects, the functional size of which was measured with the COSMIC-FFP method.

The second step was to analyze these 117 projects with respect to the ‘Data Quality Rating (DQR)’ attribute to keep only the highest quality data for statistical analysis. In the ISBSG dataset, each project has a Quality Tag<sup>5</sup> (A, B, C or D) assigned by the ISBSG reviewers, based on whether or not the data fully meet ISBSG data collection quality requirements. Considering this ISBSG recommendation, 4 of the projects with a ‘C’ or ‘D’ rating were ignored, leaving 112 projects following this filtration step.

---

<sup>5</sup> A: The data submitted were assessed as sound, with nothing identified that might affect their integrity; B: The submission appears fundamentally sound, but there are some factors which could affect the integrity of the submitted data; C: Due to significant data not being provided, it was not possible to assess the integrity of the submitted data; D: Due to one factor or a combination of factors, little credibility should be given to the submitted data.

The third step was to verify the availability of fields of size by functional type (or BFC) in the data set, for each of the 112 projects from step 2, since these fields are necessary for this study. The verification indicates that this information is not available for 20 of the projects, leaving 92 projects for the next step.

**Table 1.** Filtration of ISBSG 2007 Dataset Release10

Step	Attribute	Filter	Projects Excluded	Remaining Projects
1	Count Approach <sup>6</sup>	= COSMIC-FFP	3,989	117
2	Data Quality Rating (DQR)	= {A   B}	5	112
3	Rating for Unadjusted Function Points (UFP)	= {A   B}	20	92

Since many factors vary simultaneously, the statistical effects may be harder to identify in a more varied dataset than in a more homogeneous one. Therefore, in this study we built a series of homogeneous subsets by considering the factors which affect the size-effort relationship. We selected the project attributes that were found to significantly affect the size-effort relationship in [19][20] to build more homogeneous subsets out of the 92 remaining projects.

While exploring the nature of the relationship, the impact of other factors which were found to affect productivity, were also considered. However, many of these factors are often described, as they are in the ISBSG Repository, as categorical (or nominal) variables on which mathematical operations cannot be carried out directly. To take them into account, subsets must be built for each categorical value of such variables – referred to here as homogeneous with respect to the variable selected.

We first considered ‘Application Type’ attribute to form the homogeneous sub-datasets (Table 2- Step 4):

- Application Type
  - Management Information System
  - Financial Transaction Process/Accounting
  - Customization to a Product Data Management System
  - Others (Since the data points for this type were too few for statistical analysis, it was decided to drop them from further analysis)

Thus, in this step, 49 projects were selected and distributed among the following three homogeneous data subsets:

- Management Information System = 14 projects (all New Development Projects)
- Financial Transaction Process/Accounting = 21 projects (New Development, Re-development and Enhancement Projects)
- Customization to a Product Data Management System = 14 projects (all Enhancement Projects)

<sup>6</sup> No further filter has been considered with respect to the COSMIC-FFP versions.

**Table 2.** Further Filtration of ISBSG 2007 Dataset Release10

Step	Attribute	Filter	Projects Excluded	Remaining Projects
4	Application Type	= {Management Information System}	49	14
		= {Financial Transaction Process/Accounting}		21
		= {Customization to a Product Data Management System}		14
5	Business Type	Missing for most of the projects	-	
6	Maximum Team Size	Missing for most of the projects	-	

The next two steps (5 & 6) are to look into the availability of two other cost drivers:

- Business Type
- Maximum Team Size

However, the values of these attributes (Business Type and Maximum Team Size) are missing for most of the projects in ISBSG Release 10. Therefore, these steps were skipped.

## 4 Statistical Data Analysis and Results

The primary aim of this study is to explore whether or not an effort estimation model based on the components of functional size rather than on only a total single value of functional size would improve estimation models.

In this study, the sub-datasets formed are first analyzed to determine the strength of the relationship between the total functional size and the development effort by applying a Linear Regression Analysis method. Next, the strength of the relationship between the functional sizes of the COSMIC-FFP BFC Types used to determine total functional size and development effort is analyzed by applying a Multiple Regression Analysis method. These findings are compared to the models representing the relationship between total functional size and effort.

All the statistical data analyses in this study were performed with the GiveWin 2.10 [9] commercial tool and its sub modules.

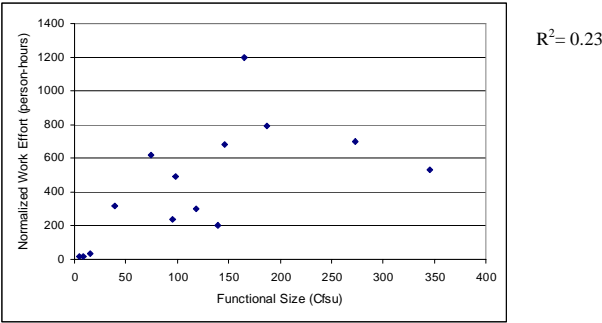
### 4.1 Total Functional Size - Effort Relationship

For the Linear Regression Analysis, the independent variable is Functional Size and the dependent variable is the Normalized Work Effort value from the Normalized Work Effort attribute. These variables are used so that the effort data among the projects that do not include all the phases of the development life cycle are comparable.

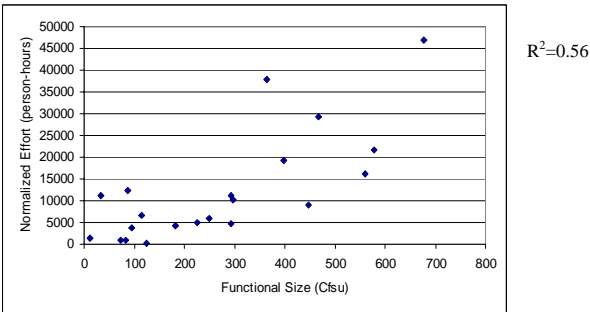
Fig. 2 shows the relationship between Normalized Work Effort and COSMIC-FFP functional size for the sub-datasets.

For the Financial Transaction Process/Accounting dataset, the  $R^2$  statistic is better than that for the Management Information Systems and Customization to a Product Data Management System datasets.

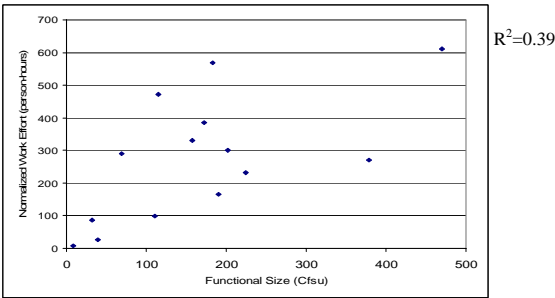
a) Sub-dataset 1: Customization to a Product Data Management System (n=14)



b) Sub-dataset 2: Financial Transaction Process/Accounting (n=21)



c) Sub-dataset 3: Management Information System (n=14)



**Fig. 2.** The Relationship between Normalized Work Effort and COSMIC Functional Size

A significance test is also carried out in building a linear regression model. This is based on a 5% level of significance. An F-test is performed for the overall model. A  $(Pr > F)$  value of less than 0.05 indicates that the overall model is useful. That is, there is sufficient evidence that at least one of the coefficients is non-zero at a 5% level of significance. Furthermore, a t-test is conducted on each  $\beta_j$  ( $0 \leq j \leq k$ ). If all the values of  $(Pr > |t|)$  are less than 0.05, then there is sufficient evidence of a linear relationship between  $y$  and each  $x_j$  ( $1 \leq j \leq k$ ) at the 5% level of significance.

The results of the linear regression analysis are given in Table 3.

**Table 3.** Regression Analysis Results (Normalized Work Effort – Total Functional Size)

Sub-dataset 1: Customization to a Product Data Management System							
Functional Size	Coeff	StdError	t-value	t-prob	Split1	Split2	reliable
$R^2 = 0.23$	3.01454	0.51622	5.840	0.0001	0.0001	0.0000	1.00000
	value	prob					
normality test	3.8843	0.1434					
Sub-dataset 2: Financial Transaction Process/Accounting							
Functional Size	Coeff	StdError	t-value	t-prob	Split1	Split2	reliable
$R^2 = 0.56$	46.61200	5.48730	8.495	0.0000	0.0000	0.0000	1.0000
	value	prob					
normality test	5.2770	0.0715					
Sub-dataset 3: Management Information System							
Constant	Coeff	StdError	t-value	t-prob	Split1	Split2	reliable
Functional Size	120.95059	69.13106	1.750	0.1057	0.0879	0.0745	0.6000
$R^2 = 0.39$	0.91522	0.33057	2.769	0.0170	0.0012	0.0080	1.0000
	value	prob					
normality test	1.9550	0.3763					

For subsets 1, 2 and 3, the Total Functional Size is found to explain about 23%, 56% and 39% of the NW\_Effort respectively.

## 4.2 Functional Sizes of BFC Types – Effort Relationship

The COSMIC-FFP method [34] is designed to measure the functional size of software based on its Functional User Requirements (FURs). In this method, each FUR is decomposed into its elementary components, called Functional Processes. A Functional Process is defined as “an elementary component of a set of FURs comprising a unique, cohesive and independently executable set of data movements” [34]. The BFCs of this method are assumed to be Data Movement Types, which are of four types; Entry (E), Exit (X), Read (R) and Write (W). The functional size of each Functional Process is determined by counting the Entries, Exits, Reads and Writes in each Functional Process. The total functional size is the sum of the functional sizes of the Functional Processes.

In this study, the Multiple Regression Analysis method [26] is used to analyze the relationship between the dependent variable Normalized Work Effort and the functional sizes of each BFC Type as the dependent variables.

The following multiple linear regression model [26] that expresses the estimated value of a dependent variable  $y$  as a functions of  $k$  independent variables,  $x_1, x_2, \dots, x_k$ , is used:

$$y = B_0 + B_1x_1 + B_2x_2 + \dots + B_kX_k \tag{1}$$

where  $B_0, B_1, B_2, B_k$  are the coefficients to be estimated from a generic data sample. The effort estimation model can then be expressed as:

$$NW\_Effort = B_0 + B_1(E) + B_2(X) + B_3(R) + B_k(W) \tag{2}$$

where  $NW\_Effort$  (Normalized Work Effort) is the dependent variable and  $E, X, R$  and  $W$  are the independent variables representing the number of Entries, Exits, Reads and Writes respectively.

In building a multiple linear regression model, the same significance tests as discussed in the previous section are carried out. Table 4 shows the multiple regression analysis results for the sub-datasets.

For subset 1, the Read category is found to explain about 41% of the  $NW\_Effort$ . For subset 2, the Entry category is found to explain 60% of the  $NW\_Effort$ . For subset 3, the Write category is found to explain 54% of the  $NW\_Effort$ .

**Table 4.** Multiple Regression Analysis Results (Normalized Work Effort – Functional Sizes of BFC Types)

Sub-dataset 1: Customization to a Product Data Management System							
	Coeff	StdError	t-value	t-prob	Split1	Split2	reliable
Read	6.69258	0.96538	6.933	0.0000	0.0000	0.0000	1.0000
$R^2 = 0.41$							
	value	prob					
normality test	2.0558	0.3578					
Sub-dataset 2: Financial Transaction Process/Accounting							
	Coeff	StdError	t-value	t-prob	Split1	Split2	reliable
Entry	220.99324	24.61603	8.978	0.0000	0.0000	0.0000	1.0000
$R^2 = 0.60$							
	value	prob					
Normality test	6.6034	0.0368					
Sub-dataset 3: Management Information System							
	Coeff	StdError	t-value	t-prob	Split1	Split2	reliable
Write	18.56507	2.08722	8.895	0.0000	0.0000	0.0000	1.0000
$R^2 = 0.54$							
	value	prob					
normality test	2.7829	0.2487					



In Table 5, the results from the two approaches are summarized.

**Table 5.** Comparison of the Results

Sub-datasets	# of Data Points	$R^2$ (Using Total Functional Size (CFP))	$R^2$ (Using BFC Types)
Sub-dataset 1	14	0.23	0.41
Sub-dataset 2	21	0.56	0.60
Sub-dataset 3	14	0.39	0.54

We further investigated the functional profiles of the projects in the sub-datasets (see Table 6).

A functional profile provides information about the distribution of functionality within specific software, i.e. the percentages of Entries, Exits, Reads and Write in COSMIC FFP and permits comparison of its functional distribution with that of a sample of projects its for which the average functional profile is known [4].

**Table 6.** Functional Profiles of the Projects in the Sub-datasets

a) Sub-dataset 1: Customization to a Product Data Management System

	Entry (%)	Exit (%)	Read (%)	Write (%)
Max	<b>40%</b>	<b>36%</b>	<b>60%</b>	<b>20%</b>
Average	32%	14%	47%	7%
Median	34%	11%	46%	6%
Min	<b>13%</b>	<b>2%</b>	<b>37%</b>	<b>2%</b>

b) Sub-dataset 2: Financial Transaction Process/Accounting

	Entry (%)	Exit (%)	Read (%)	Write (%)
Max	<b>50%</b>	<b>54%</b>	<b>48%</b>	<b>33%</b>
Average	22%	35%	29%	14%
Median	18%	36%	33%	12%
Min	<b>9%</b>	<b>3%</b>	<b>0%</b>	<b>0%</b>

c) Sub-dataset 3: Management Information System

	Entry (%)	Exit (%)	Read (%)	Write (%)
Max	<b>56%</b>	<b>74%</b>	<b>24%</b>	<b>17%</b>
Average	39%	43%	10%	8%
Median	40%	42%	8%	7%
Min	<b>18%</b>	<b>26%</b>	<b>0%</b>	<b>0%</b>

## 5 Discussion and Conclusions

This study has explored whether an effort estimation model based on the functional sizes of BFCs Types rather than the total functional size value would provide better results. Our hypothesis was that the development effort for each of the BFC Types, which provide different user functionalities, might be different.

For the statistical analyses, the dataset of ISBSG 2007, Release 10 [10], was used. Projects measured by COSMIC FFP were selected. While exploring the nature of the relationship between functional size and effort, some factors which are found to most affect the size-effort relationship were also taken into account. At the end of the filtration process, three homogeneous subsets of projects are built based on the Application Types of the projects and with enough data points for statistical analysis.

The  $R^2$  statistics were derived from Linear Regression Analysis to analyze the strength of the relationship between total functional size and normalized work effort. The results were compared to the  $R^2$  statistics derived from the Multiple Regression Analysis performed on the Functional Sizes of the BFC Types and Normalized Work Effort.

Increases in  $R^2$  values (0.23 to 0.41 for Sub-dataset 1; 0.56 to 0.60 for Sub-dataset 2 and 0.39 to 0.54 for Sub-dataset 3) were observed when the functional sizes of each of the BFC Types are taken into account for effort estimation purposes instead of the total functional size. The results showed a significant improvement in the modeling of the size-effort relationship in the estimation models for at least two of the subsets.

An interesting observation in this study is that the functional size of a single BFC Type, i.e. Reads in Sub-dataset 1, Entries in Sub-dataset 2 and Writes in Sub-dataset 3, can model Normalized Work Effort.

We also analyzed the dominating functionality types (Entry, Exit, Read, Write) in each of the three samples (e.g. with the greatest frequency distribution). For Sub-dataset 1, it is the Read (46 %) and Entry (34%) function types that are dominant among the four BFC types. For Sub-dataset 2, Exit (36%) and Read (33%) are both dominant, while for Sub-dataset, it is the Entry (40%) and Exit (42%) BFC types those are dominant.

Although, for Sub-dataset 1, the dominating BFC Types, i.e. Reads can model the Normalized Work Effort, for the other Sub-datasets, we could not find a relationship with the dominating BFC types and the BFC Types which can model the work effort.

Our hypothesis in this study was developing different functionality types requires different amounts of work effort. Therefore, a probable estimation model might involve different weights for each BFC Type.

The results of this study indicate that more research is needed to analyze the effect of different BFC Types on effort estimation. The effort required to develop software for different functional domains might be better explained by taking into account the functional sizes of different BFC Types. Therefore, not only the evidence about possible dominating BFC types per certain sub-datasets, but the level of contribution of each of the BFC Types to work effort needs to be investigated more in the near future. Further work should also include comparisons with related work performed with the FPA method.

## References

- [1] Albrecht, A.J.: Measuring Application Development Productivity. In: Proc. Joint SHARE/GUIDE/IBM Application Development Symposium, pp. 83–92 (1979)
- [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., Gil, B., Lefebvre, E.: Estimation Models Based on Functional Profiles. In: International Workshop on Software Measurement – IWSM/MetriKon, Kronisburg, pp. 195–211. Shaker Verlag, Germany (2004)
- [4] Abran, A., Panteliuc, A.: Estimation Models Based on Functional Profiles. III Taller Internacional de Calidad en Tecnologías de Información et de Comunicaciones, Cuba, February 15–16 (2007)
- [5] Boehm, B.W.: Software Engineering Economics. Prentice-Hall (1981)
- [6] 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)
- [7] Gencel, C.: An Architectural Dimensions Based Software Functional Size Measurement Method, PhD Thesis, Dept. of Information Systems, Informatics Institute, Middle East Technical University, Ankara, Turkey (2005)
- [8] Gencel, C., Demirors, O.: Functional Size Measurement Revisited. Scheduled for publication in ACM Transactions on Software Engineering and Methodology (2007)
- [9] GiveWin 2.10, <http://www.tsprintl.com/>
- [10] ISBSG Dataset 10 (2007), <http://www.isbsg.org>
- [11] 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)
- [12] Jeffery, R., Ruhe, M., Wiecezorek, 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)
- [13] Jørgensen, M., Molokken-Ostfold, 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)
- [14] Kitchenham, B., Mendes, E.: Software Productivity Measurement Using Multiple Size Measures. IEEE Transactions on Software Engineering 30(12), 1023–1035 (2004)
- [15] Briand, L.C., El Emam, K., Maxwell, K., Surmann, D., Wiecezorek, 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, CA, USA, pp. 313–322 (1998)
- [16] Briand, L.C., Langley, T., Wiecezorek, 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)
- [17] Menzies, T., Chen, Z., Hihn, J., Lum, K.: Selecting Best Practices for Effort Estimation. IEEE Transactions on Software Engineering 32(11), 883–895 (2006)
- [18] Leung, H., Fan, Z.: Software Cost Estimation. Handbook of Software Engineering, Hong Kong Polytechnic University (2002)
- [19] 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)

- [20] Forselius, P.: Benchmarking Software-Development Productivity. *IEEE Software* 17(1), 80–88 (2000)
- [21] Lokan, C., Wright, T., Hill, P.R., Stringer, M.: Organizational Benchmarking Using the ISBSG Data Repository. *IEEE Software* 18(5), 26–32 (2001)
- [22] Maxwell, K.D.: Collecting Data for Comparability: Benchmarking Software Development Productivity. *IEEE Software* 18(5), 22–25 (2001)
- [23] 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)
- [24] Naur, P., Randell, B. (eds.): *Software Engineering, Conference Report*, NATO Science Committee, Garmisch, Germany, 7–11 October (1968)
- [25] 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* (2005)
- [26] Neter, J., Wasserman, W., Whitmore, G.A.: *Applied Statistics*. Allyn & Bacon (1992)
- [27] ISO/IEC 14143-1: *Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts* (1998)
- [28] ISO/IEC 14143-1: *Information Technology – Software Measurement – Functional Size Measurement – Part 1: Definition of Concepts* (February 2007)
- [29] 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)
- [30] ISO/IEC TR 14143-3: *Information Technology – Software Measurement – Functional Size Measurement – Part 3: Verification of Functional Size Measurement Methods* (2003)
- [31] ISO/IEC TR 14143-4: *Information Technology – Software Measurement – Functional Size Measurement - Part 4: Reference Model* (2002)
- [32] 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)
- [33] ISO/IEC 14143-6: *Guide for Use of ISO/IEC 14143 and related International Standards* (2006)
- [34] ISO/IEC 19761:2003, *Software Engineering – COSMIC-FFP: A Functional Size Measurement Method*, International Organization for Standardization (2003)
- [35] ISO/IEC 20926:2003, *Software Engineering-IFPUG 4.1 Unadjusted Functional Size Measurement Method - Counting Practices Manual*, International Organization for Standardization (2003)
- [36] ISO/IEC 20968:2002, *Software Engineering – MK II Function Point Analysis – Counting Practices Manual*, International Organization for Standardization (2002)
- [37] 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)
- [38] 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)
- [39] The International Function Point Users Group (IFPUG). *Function Points Counting Practices Manual* (release 4.2), International Function Point Users Group, Westerville, Ohio (January 2004)

- [40] United Kingdom Software Metrics Association (UKSMA). MkII Function Point Analysis Counting Practices Manual, v 1.3.1 (1998)
- [41] The Common Software Measurement International Consortium (COSMIC). COSMIC-FFP v.2.2, Measurement Manual (January 2003)
- [42] The Common Software Measurement International Consortium (COSMIC). COSMIC-v.3.0, Measurement Manual (September 2007)

# Experiences on Using Software Experiments in the Validation of Industrial Research Questions

Dominik Gessenharter<sup>1</sup>, Alexander-Marc Merten<sup>2</sup>, Alexander Raschke<sup>1</sup>,  
and Nicolas Fernando Porta<sup>2</sup>

<sup>1</sup> Ulm University, Institute of Software Engineering  
and Compiler Construction, D-89069 Ulm, Germany

`Dominik.Gessenharter@uni-ulm.de`, `Alexander.Raschke@uni-ulm.de`

<sup>2</sup> Daimler AG, Research & Development, Ulm  
`alexander-marc.merten@daimler.com`, `nicolas.porta@daimler.com`

**Abstract.** Experimentation in software engineering is difficult. One reason is the large number of context variables [1] and the impracticality of experiments in an industrial setting. Considering the budgets of comprehensive projects, it is apparent that a company cannot double its effort executing a project twice, in order to compare two different approaches concerning process or method improvement. Performing experiments on the basis of small projects seldom offers solutions valid for industrial settings. Our commendation is a cooperation between industry and academic education. This approach offers multiple advantages. In this paper, we outline our experiences in experimental software engineering gained in about 20 experiments over the past 10 years by the Ulm University cooperating with Daimler AG, Research & Development, Ulm. Additionally we provide an insight into a current experiment and present our approach to experimental software engineering in further detail.

## 1 Introduction of an Cooperation of Industrial and Academic Educational Stakeholders

The cooperation of the Ulm University and Daimler implements the idea of proving new approaches concerning software or process engineering and improvement. Whereas Daimler describes the way of looking at a problem and proposes a solution, the University proves the proposal by conducting an applicable experiment. Though, the crux of the matter is, that academic and industrial environments have little deals in common. In addition, the real world and the surroundings of an experiment are in principle two different things. In spite of these differences, the described cooperation is a win-win situation for both partners as we will show.

### 1.1 Ulm University and Daimler AG Group Research and Advanced Engineering - Stakeholders of Two Different Application Areas

The main interest of an industrial stakeholder is to have a prove of concept for ideas in improving software engineering or software development processes. In

order to benefit from innovations in the software engineering processes or the rise of product quality by using new tools, innovations or tools are used in a test run at the university. This task is accomplished by offering a practical course for students who will sample the modified process or the tool that might - after further research - be introduced. Whereas companies take stock in the results of such a field test, the university is interested in current industrial practises.

## **1.2 The Idea of Merging Industrial and Academic Issues**

There is a win-win situation for both, the industrial and the academic part of the cooperation. On the one hand, the industrial partner is outsourcing the planning, conduction and evaluation of the experiments and the measured data. That leads to considerable reduce of costs on the industrial side. This advantage is not paid by less quality rather than by having impartial participant for the experiment, i.e. the students. They do not have any knowledge about customs or preferences in the business practice of the industry in general or the actual cooperation partner in detail. However, the lack of experience of the student in contrast to the staff of a company must be considered. On that account, the subject matter of an experiment is not influenced by any habits of the assumed domain.

On the other hand, the university obtains the possibility to teach their students nowadays business practise and problems.

## **1.3 Empirical Software Engineering vs. Experimental Software Engineering**

Empirical Software Engineering makes use of evaluating data collected from any project that is related to the matter of interest. The problem is, that collecting data in some projects lacks the comparability if there are no data to be compared to. When gathering data from experiments, it is possible to change the surrounding variables to figure out a benefit or a drawback on any modification of the software engineering process, methodologies or the used tools. Furthermore projects in industrial environment can not take the risk of implementing methods without having any idea of their practicability as the financial impact in case of a failure would be unacceptable.

We consider our efforts to be rather experimental software engineering than empirical software engineering.

## **1.4 The Set-Up of a Typical Experiment**

An experiment needs a planning, conduction and evaluation phase. Every phase is essential since every deficiency in any phase might cause following deficiencies in subsequent phases. Bad planning as for example bad scheduling, will likely result in pressure of time during some parts of the experiment and therefore the collected data may be not feasible. In the following three chapters, we describe our proceeding and our experiences of each phase in experimental software engineering.

## 2 Planning, Scheduling and Designing an Experiment

With this chapter, we describe an experiment as it is typical in our cooperation with Daimler. This includes details of the experiments object, its scheduling and some aspects of how teamwork can be organized.

### 2.1 Planning the Product, Development-Phases, Development-Process and Deliverables

Experiments as we did for the past 10 years are always used to support the evaluation of hypotheses which deal in most cases with process or method improvements. The experiment therefore must be based on convenient artefacts and use adequate processes. Cooperating with Daimler, we most often use examples of the field of embedded systems, e. g. different kinds of controllers in a car whereas processes are often varied.

When being committed to a product, the next step in planning an experiment is to decide, which phases of the development process are necessary for the analysis of the experiments hypothesis. In some cases, we concentrate on the analysis phase and typical documents like a specification of a product, in other cases design phase or exclusively the implementation are the focus of interest.

Depending on the hypothesis, the observed phases and the underlying process as well as all deliverables must be specified and documented. The reason is, that the experiment must be comprehensible and controllable. Therefore, every detail of the planning of an experiment must be available in written form in order to be able to iterate the experiment. A documentation of every single step in planning and conduction is a crucial point whenever the product or document quality is used as an indicator of the compliance of a hypothesis.

### 2.2 Scheduling of an Experiment

The main focus when scheduling an experiment is to keep in mind, that the overall time must be limited. If not, the experiment is to distant from reality, where time is an economic factor. Due to the fact that students are not experienced developers, a training phase should be arranged before an experiment starts. The remaining overall time must be divided into fixed periods assigned to special activities. The partition of the development process is the basis of a division of work that allows to simulate a client-contractor relationship. Therefore, the deadlines for deliverables of each phase must be fixed at the very beginning of the experiment. During training, all examples should not antedate problems or even solutions of the experiments object.

### 2.3 Designing an Experiment

We consider the design of an experiment as a means of controlling how students work together. On the one hand, we support team work all over the experiments duration. On the other hand, a team must not consist of too many students.



The reason is obvious: The students acting as inexperienced developers work on a project that can be handled within several weeks or months. Therefore, the experiments object must be clear and manageable under the given conditions. Teams with too many members often have unbalanced workload on the team members. As far as our experience goes, a team with more than two members has at least one member with less work than the others.

Another reason for small teams is, that more teams can be built. We aim to build groups that consist of teams. Each group has as many teams as the development process has phases. This allows us, to simulate different departments each taking on responsibility for one specific phases of the same development process. If the number of participants in an experiments suffices, multiple groups are set up.

The motivation for building teams is to balance workload on the one hand and to make teams customer or suppliers of / for other teams. The output of a team in  $phase_i$  is the input for another team in  $phase_{i+1}$ .

Having multiple groups, the results are more reliable. Side-effects can be filtered by comparing the groups' results. Another option is to use one group as a comparison group that is working in a common setup whereas another group is working under conditions in line with the experiments hypothesis. The crux of the matter is, that it is not trivial to outline what effects are a result of different setups when having a comparison group and not the result of difference between the members of the different groups. To draw conclusions about success or failure of the experiments object, it is necessary to have the right data and metrics when evaluating the experiment. Both aspects are discussed in the next two chapters.

### 3 Conduction of an Experiment

With this chapter, we describe how the experiment should be controlled. Therefore, we focus on variables influencing the experiment and providing data for later measurement, opinions of participants as an hardly gaugeable variable and the means of collecting data.

#### 3.1 Controlling the Experiment

Controlling an experiment is difficult. Actions taken in order to control an experiment usually lead to a change of variables. This might cause unexpected side-effects. The main problem is to determine, whether there have been side-effects and if so, to trace them appropriately.

An adaption of the project scheduling is one of the easiest ways to allow for planning deficiencies. Whenever the time need for a phase is underestimated or overestimated, a following phase can be shifted. Therefore, it is necessary to have some spare time that can be used as a buffer. If adoptions of the scheduling is not possible, another possibility is to reduce work by reduction of the experiments' object complexity.

Being interested in the impact of process changes or tool exchange we always try not to change anything else than the projects schedule or complexity.

Whenever this constraint in controlling is too constrictive, a wider combination of responses to unexpected discrepancies is required. But this entails the side-effect problem. Accordingly, the planning of an experiment cannot be more than adequate since a proper planning is the most promising way of keeping actions for controlling on a minimum level.

### 3.2 Collecting Data

The result of an experiment depends on the data collected during its conduction. Bad data lead to non-significant or false interpretations, even if all other aspects of planning and controlling the experiments have been considered.

In order to get good data, we make use of different data sources. This way, we can compare results of one source with those of another one, e.g. development time spent on meetings can be determined by meeting protocols or by students' time reporting. The next step is to decide, which data are good. When having much data, errors in measurement can be levelled. But there is a high risk of wrong decisions. On the one hand, we have to accommodate some fuzziness, on the other hand, we try to border this fuzziness the best we can. Therefore, we have no standard technique yet but try to get an understanding of data and measurement errors or misinterpretation by looking on discrepancies of data from different sources and discussing possible reasons.

### 3.3 Objective Data

Objective data are data that are impartial. They can be captured by measuring, e.g. the expenditure of time (measured in person hours) or the product quality as the degree of performance (by counting correctly implemented requirements). It is not always easy to get these data, but in most cases, it is possible.

In the last experiment, we used a special infrastructure to log all e-mails between participants of the experiment. By counting the characters of all e-mails, an objective indicator of communication quantity via e-mail is found. Although the information content or the time of typing the email is unknown and therefore neither the effectiveness nor the time need of communication is numbered, communication patterns during the project phases can be traced.

### 3.4 Subjective Data

Using questionnaires is a way of getting subjective data. These data may be far from reality. But a combination of objective and subjective data can be used to calibrate subjective data. In most experiments, we asked for data that could be achieved in an objective way, too. The detected bias can not be transferred to other data to which an objective measurement is not available, but it helps to estimate the accuracy of the data collected via questionnaires. Furthermore, it provides an insight to the perception of the participants.

### 3.5 Opinions and Feelings

Opinions and feelings are often unsuitable to be a basis for interpretation of an experiments result. Especially in our particular surroundings where students substitute experienced developers.

However, opinions can be hints to method deficiencies and provide valuable insights. Though this is no hard data, it helps in improving ideas and foster further research. Moreover, keeping in touch with the participants has proven to be an important factor in controlling and steering an experiment. We therefore recommend the use of questionnaires and feedback rounds in a mid- to long-term experiment as often as it is appropriate. We suggest one feedback round per week, one questionnaire per experiment phase and one or two intensive discussion rounds per experiment (middle and end).

### 3.6 Introduction of a Sample Experiment

In the following chapter we will discuss our latest experiment concerning the evaluation of user-defined workflow (UDWF) [5] concepts. UDWF is a concept to support self-organizing team work in distributed development environments. In order to coordinate work, workflow and working structures are modelled by the users themselves, without the need for extensive knowledge of workflow theory. To a certain extent, this can be achieved by the use of standard office software, also called informal coordination. The aim of the experiment was, to create a data baseline concerning informal coordination, which can then in turn be used to evaluate tool-supported UDWF concepts in a second experiment. By comparing the data of both experiments, we want to achieve new insights in the effectiveness of UDWF concepts and prove or refute research ideas concerning the support of team work. To achieve a maximum of validity we will only change the way teamwork is supported. Neither process, nor the deliverables of the experiment will be varied.

## 4 Analysis and Evaluation of the Experiment

In order to measure the effect of UDWF-concepts the following actions have been taken:

- definition of the project goals
- identification of the benefit-indicators and formulation of measurement-hypothesis
- definition of cause-and-effect-chains
- definition of metrics
- collection of metric data
- interpretation of the collected data

For project goals we chose *overall development time*, *overall development cost* and *product quality*. In the definition of the corresponding cause-and-effect-chains we identified the following benefit-indicators:

- process constancy
- coordination effort
- changeability
- development effort
- product quality
- tool acceptance

The impact of UDWF concepts on the project goals concerning time, cost and quality were assessed. The effects which start with the initiative introduction of UDWF concepts were traced over benefit-indicators through the cause-and-effect-chain finally influencing the project goals (for a detailed insight in the cause-and-effect-chains and benefit indicators see [4]).

On this basis, taking into account the qualitative relations between the benefit indicators themselves and between benefit indicators and project goals, metrics were defined in order to quantify these relations. Figure 1 shows the metric meeting time as an example.

<b>Definition</b>	Man hours for meetings, summed up for each phase and each team and per person.
<b>Unit of measurement</b>	Man hours (example: in case of 4 persons meeting for 2 hours, this is 8 man hours)
<b>Interpretation</b>	More meeting hours means more coordination effort
<b>Problems</b>	Line between coordination and development effort not always clear. Meetings can be used either to discuss or to develop or for both.
<b>Associated Goal</b>	Reduction of (superfluous) coordination effort
<b>Data source</b>	Meeting protocols
<b>Method of data collection</b>	Data is collected from meeting protocols by hand
<b>Measuring frequency</b>	Once per week
<b>Remarks</b>	

**Fig. 1.** A sample metrics for time spend on meetings

The template is an adapted version of [2]. It contains a table to define the metric and add information about the source used for data collection and measurement. According to the principles of Goal Question Metric [3] the intended interpretation of a metric should be known from the beginning. The example "meeting time" can be interpreted in the way "the bigger the meeting time measured the bigger the coordination effort". At the same time risk and problems should be documented. In this case one risk is the difficult separation between coordination and development effort. Finally the measurement schedule is documented. In this example the data for the metric is collected every meeting (assuming that the students have several meetings a week).

An overview of all captured metrics is provided in figure 2.

Analyzed Object	Metrics	Resolution			Remark
		Phase	Group	Team	
Meeting hours	#man hours	x	x	x	Meeting hours are part of the coordination effort
	#meeting hours / person				Answers the question on how much time a participant spent on the average in meetings
E-Mails	#characters in all E-Mails	x	x	x	Signitures and citings have been deleted byscript and counter checked byhand.
	#standard-E-Mails	x	x	x	The standard length of a email is the median over all emails tracked. This is used to divide the overall number of characters to get the number of standard emails.
Development plan	#weighted development plan entries	x	x	x	Entries without comments and additional hints are awarded two points. Entries with additional hints are awarded three points
	# weighted entries / #persons				Shows how manyentries a person has written in the development plan on the average.
Functionalität (scope of implementation)	#SLOC		x		Scope of implementation in lines of code
	#SLOC (planned functions)				Scope of implementation for planned functions
	#SLOC (changes)				Scope of implementation for unplanned functions and specification changes
Development effort	#man hours	x	x	x	Effort invested into development activities
Coordination effort	#man hours	x	x	x	Effort invested into coordination activities
Friendliness towards introduced changes	#SLOC (planned functions) / #SLOC (changes) per Group				Here we compare lines of code for planned features against lines of code for changes. This ratio is used to determine the friendliness towards change.
Productivity	#SLOC / (development effort + coordination effort)	x	x		Productivity as the result of the scope of implementation divided by the overall project effort
Quality (developer test)	#errors		x		Quality of the product before developer tests are conducted.
	#critical errors		x		
	#testcases		x		
Quality (acceptance test)	#errors		x		Quality of the final product. Supervisor evaluate the products of each group using the same test cases.
	#critical errors		x		

Fig. 2. List of all used metrics

What are the conclusions that can be derived by the metrics? As our research goal we want to assess whether informal coordination, by means of Open-Office documents, can be improved by the use of tool-supported coordination, specifically designed to address UDWF issues.

Hypothesis:

Effort (informal coordination) > Effort (tool-supported coordination)

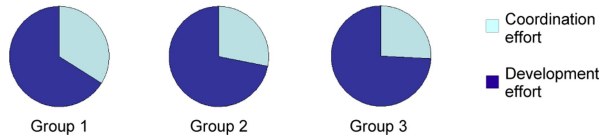
Alternative hypothesis:

Effort (informal coordination)  $\leq$  Effort (tool-supported coordination)

In this first experiment the relation between development and coordination effort we obtained was quite similar in all groups: constantly between 1:3 and 1:4 (figure 3). With this baseline we can asses the coordination effort in further experiments and search for proves of our hypothesis.

In addition to the metrics, questionnaires were used to evaluate those benefit-indicators which are difficult to measure like process constancy. Furthermore they support the justification of metrics. E.g. we looked at the e-mail volume. It only makes sense to analyze the written e-mails if e-mail communication is used as a tool for coordination (which has been done in fact - the students answered they had used e-mails as most frequent media for coordination after personal talks).

With the well-defined set of metrics in combination with questionnaires we can support the right interpretation of the obtained data and clearly outline the differences between the diverse processes, methods or tools we focus on in the experiments.



**Fig. 3.** Ratio of coordination effort to development effort of the three groups

## 5 Conclusion and Future Work

We described how a cooperation between an industrial and an academic partner can lead to a win-win situation for both parts. It offers access to business practices for students and academic researchers as well as to reduction to companies. The results are not scaled just as costs are reduced. Nevertheless, controlling an experiment is difficult due to the different surroundings when being conducted at an university and not under industrial settings. We discussed some approaches on how to deal with these problems, although more research is required on this topic.

Some future work will be to look for answers to the question how experiments in academic surroundings can produce more reliable results for industrial settings as well as a closer look on and better understandings of variables and dependencies among them.

## References

1. Basili, V.R., Shull, F., Lanubile, F.: Building Knowledge through Families of Experiments. *Software Engineering* 25, 456–473 (1999)
2. Kowitz, A., Ofer, C.: Durchführung eines Messprogramms: ein Erfahrungsbericht. In: Abran, A., Bundschuh, M., Büren, G., Dumke, R.R. (eds.) *Applied Software Measurement: Proceedings of the International Workshop on Software Metrics and DASMA Software Metrik Kongress IWSM/MetriKon 2006*, pp. 455–471. Shaker, Aachen (2006)
3. Basili, V., Caldiera, G., Rombach, H.D.: Goal Question Metric Approach, *Encyclopedia of Software Engineering*, pp. 528–532. John Wiley & Sons, Inc. (1994)
4. Porta, N.F., Merten, A.-M.: Konzepte zur Kosten-Nutzen-Messung prozessverbessernder Manahmen in der Automotive E/E Entwicklung am Beispiel User Defined Workflow, *Metrikon* (2007)
5. Merten, A.-M.: User Defined Workflow (UDWF) - Grundlagen und Umsetzungsaspekte im Anwendungsbereich Forschung und Entwicklung (Master Thesis), Universität Ulm (2006)
6. Houdek, F.: Empirisch basierte Qualitätsverbesserung - Systematischer Einsatz externer Experimente im Software Engineering. Dissertation, Logos Verlag Berlin (1999) ISBN 3-89722-299-X

# How to Measure Agile Software Development

Martin Kunz<sup>1</sup>, Reiner R. Dumke<sup>1</sup>, and Andreas Schmietendorf<sup>2</sup>

<sup>1</sup>Software Engineering Group

University of Magdeburg, Germany

{makunz,dumke}@ivs.cs.uni-magdeburg.de

<sup>2</sup>Berlin School of Economics, Germany

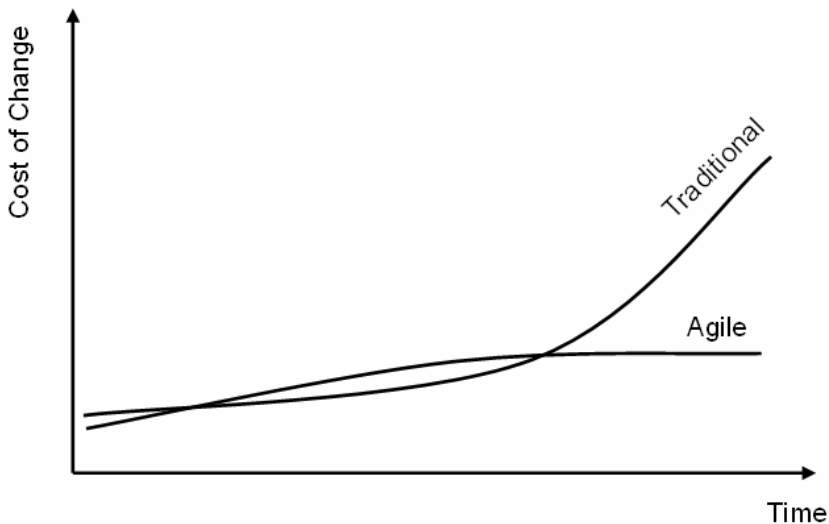
schmiete@fhw-berlin.de

**Abstract.** Agile Software Development Methods are nowadays wide spread and accepted. From the Software Measurement point-of-view not all metrics and methods from conventional lifecycle models can be used without adoption. Therefore this paper describes distinct metrics and their implementation into a measurement tool for quality management in agile software development. Taking a closer look to agile methods we observe that some of metrics and the tool itself can be used for measurement in traditional software development as well.

**Keywords:** Software Measurement, Agile Software Development, Metrics.

## 1 Introduction

Many technological ambitious products were designed with new complex functionality. The demand for functions establishes a need for new software requirements to deliver new functionality. Due to the fast alteration and the high cost of change in the



**Fig. 1.** Cost of Change compared to development method [Beck1999]

late life cycle phases the agile software development method becomes more important in this field of application. Agile software development methods like eXtreme programming try to decrease the cost of change and therewith reduce the overall development costs.

The different cost of change for agile software development in comparison with traditional software development according to the project progress is shown in Figure 1.

To realize this totally different project characteristic, the agile software development has established a different type of product life cycle in comparison with traditional life cycle models like waterfall-model or V-Model.

Extreme Programming (XP) is a usual and wide spread agile software development method. Therefore, we have used XP as a reference model for agile software development.

The major element of the XP life cycle is the “Iteration”. The iteration is recurring event in which an actual version is edited for example by adding additional functionality, correcting errors or removing unnecessary functionality. The result of each iteration is validated through an acceptance test.

In XP the duration of each iteration is very short (1-4 weeks) in comparison with traditional software development. So the planning phase is also very short and mainly realized by the definition of tasks.

The general characteristics of the XP life cycle are shown in Figure 2.

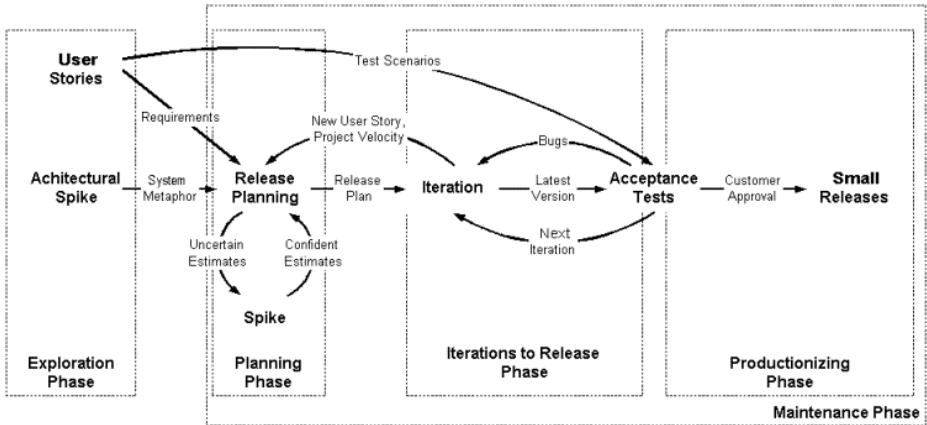


Fig. 2. eXtreme Programming life cycle [Wells2007]

Besides the different life cycle, agile software development is distinguished by one major element: the refactoring.

Refactoring means the change of source-code and software-design without altering the functionality or external behavior of software [Beck1999]. In XP, a refactoring cycle is proposed after each iteration to adjust the software product. The main goal is to ensure product quality attributes [Fowler2000].

Therefore the refactoring cycle has essential importance for agile software development, so it is a fundamental toehold to support agile software development.



## 2 Product Metrics for Agile Software Development

To support agile software development and especially refactoring, mainly source-code based product metrics are beneficial to increase quality and productivity [Goodman04].

Primarily internal quality attributes have to be ensured to control the source-code quality and to evaluate refactoring steps [McConnell2004].

The goal of our approach is to combine refactoring with software measurement to give advice about the following aspects:

- Appropriate point in time for necessary refactoring steps
- Significance of a refactoring step and the apparent quality effect
- Side effects of refactoring steps

The metrics should deliver indices for distinct refactoring steps and they should be easily interpretable. The measurement results should be a trigger or activator for useful refactoring steps.

There are various possibilities to define a set of metrics for a distinct information need or for specific conclusions. A lot of different approaches were used for example ISO/IEC 9126 [ISO/IEC2001] or the Goal Question Metric-Method [Basili19984]. Additionally, we used metrics and calculation methods out of agile environment [McConnell2004] [Henderson1996] [Fowler2000].

One major base was the metrics suite from Chidamber & Kemerer [Chidamber1993]. Secondly, we chose source code metrics among others: Number of Name-Parts of a method (NNP), Number of Characters (NC), Number of Comment-Lines (CL), Number of Local Variables (NLV), Number of Created Objects (NCO), and Number of Referring Objects (NRO). [Henderson1996][Bloch2001]

For the interpretation and analysis we normalize the value according to different thresholds.

The mapping between the normalized values and the linguistic quality terms are shown in the following table:

**Table 1.** Mapping between values and linguistic terms

Linguistic term	Normalized value
Best/Excellent	1.00
Very Good	0.82
Good	0.66
Average	0.50
Poor	0.32
Very Poor	0.16
Worst	0.00

The thresholds for the normalization are defined on the basis of empirical data but they can easily be modified according to distinct projects specifications.

### 3 “UnitMetrics” – Measurement Tool

To support agile software development at the origin we decided to implement the tool as a Plug-In for Integrated Development Environments (IDE). We choose Eclipse because of the preconditions and its open source character [Eclipse].

The general goal was to create a measurement tool which expands the Eclipse-Development-Environment to provide source-code analysis by the use of appropriate metrics.

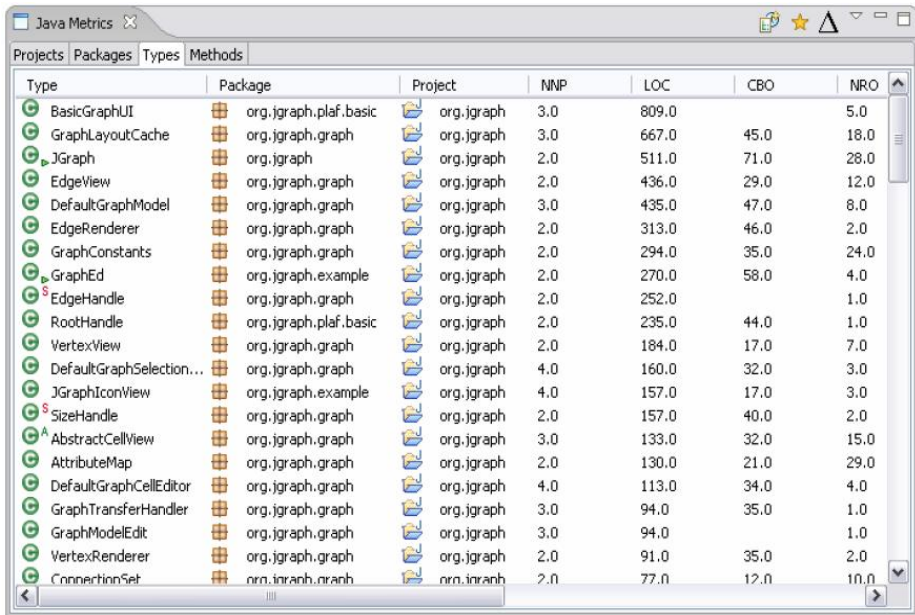
Four major functionalities have to be implemented:

- Continuous analysis
- Fundamental interpretation
- Interactive visualization
- Extensibility

Another important feature is the implementation of a snapshot concept. It is an important concept for the usage of the tool because it allows comparing the actual values with previous ones. In this way it makes it possible to make conclusions about product changes.

Especially XML-based import and export of snapshots provides a lot of additional possibilities for analyzing measurement data.

In Figure 3 one can see the measured values for a chosen snapshot. The metrics are calculated at method level and without normalization.



Type	Package	Project	NNP	LOC	CBO	NRO
BasicGraphUI	org.jgraph.plaf.basic	org.jgraph	3.0	809.0		5.0
GraphLayoutCache	org.jgraph.graph	org.jgraph	3.0	667.0	45.0	18.0
JGraph	org.jgraph	org.jgraph	2.0	511.0	71.0	28.0
EdgeView	org.jgraph.graph	org.jgraph	2.0	436.0	29.0	12.0
DefaultGraphModel	org.jgraph.graph	org.jgraph	3.0	435.0	47.0	8.0
EdgeRenderer	org.jgraph.graph	org.jgraph	2.0	313.0	46.0	2.0
GraphConstants	org.jgraph.graph	org.jgraph	2.0	294.0	35.0	24.0
GraphEd	org.jgraph.example	org.jgraph	2.0	270.0	58.0	4.0
EdgeHandle	org.jgraph.graph	org.jgraph	2.0	252.0		1.0
RootHandle	org.jgraph.plaf.basic	org.jgraph	2.0	235.0	44.0	1.0
VertexView	org.jgraph.graph	org.jgraph	2.0	184.0	17.0	7.0
DefaultGraphSelection...	org.jgraph.graph	org.jgraph	4.0	160.0	32.0	3.0
JGraphIconView	org.jgraph.example	org.jgraph	4.0	157.0	17.0	3.0
SizeHandle	org.jgraph.graph	org.jgraph	2.0	157.0	40.0	2.0
AbstractCellView	org.jgraph.graph	org.jgraph	3.0	133.0	32.0	15.0
AttributeMap	org.jgraph.graph	org.jgraph	2.0	130.0	21.0	29.0
DefaultGraphCellEditor	org.jgraph.graph	org.jgraph	4.0	113.0	34.0	4.0
GraphTransferHandler	org.jgraph.graph	org.jgraph	3.0	94.0	35.0	1.0
GraphModelEdit	org.jgraph.graph	org.jgraph	3.0	94.0		1.0
VertexRenderer	org.jgraph.graph	org.jgraph	2.0	91.0	35.0	2.0
ConnectionSet	org.jgraph.graph	org.jgraph	2.0	77.0	12.0	10.0

Fig. 3. Measurement values for a snapshot [Kersten2007]

One has to recognize that without normalization the interpretation is very difficult due to less clarity.

To give the user a more graphical representation of the normalized values we established a “star” concept where three stars means excellent and zero stars means worst ( in half star steps according to the mapping in table 1).

Figure 4 shows the same results as shown in Figure 3 in a normalized way by using the “star” concept.

Type	Package	Project	NNP	LOC	CBO	NRO
BasicGraphUI	org.jgraph.plaf.basic	org.jgraph	☆☆☆	☆☆☆		
GraphLayoutCache	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
JGraph	org.jgraph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
EdgeView	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
DefaultGraphModel	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
EdgeRenderer	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
GraphConstants	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
GraphEd	org.jgraph.example	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
EdgeHandle	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
RootHandle	org.jgraph.plaf.basic	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
VertexView	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
DefaultGraphSelection...	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
JGraphIconView	org.jgraph.example	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
SizeHandle	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
AbstractCellView	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
AttributeMap	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
DefaultGraphCellEditor	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
GraphTransferHandler	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
GraphModelEdit	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
VertexRenderer	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	
ConnectionSet	org.jgraph.graph	org.jgraph	☆☆☆	☆☆☆	☆☆☆	

Fig. 4. Normalized measurement values [Kersten 2007]

Another important set of views is represented by dependence graphs [Henderson1996] [McConnell2004]. A dependence graph displays the coupling between different elements. At the moment the visualization shows the coupling on type-(class) - and package-level.

Dependence graphs allow to display the netting of coupling and to identify strong components. A strong component contains all elements which are connected into a loop. Because of the fact that a goal of agile software development is to avoid loops or circles, an identification is very useful.

Figure 5 shows an example of a dependence graph on package level. As one can see nearly every package is connected to every other package which manifests a loop-problem.

Figure 6 shows the dependences on type-level and allows the further analysis of the discovered loops. At this point it stands out that the loops are not limited to interfaces but also obtain other components.

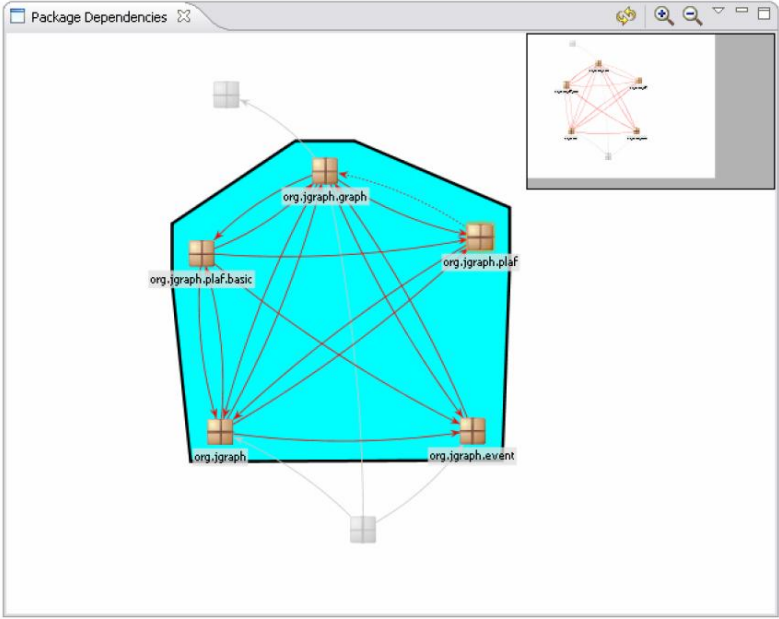


Fig. 5. Dependence graph on Package-Level

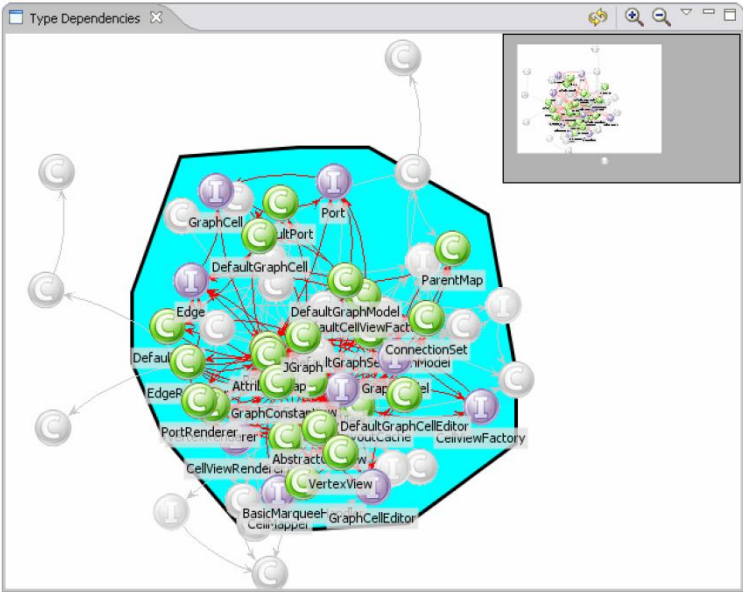


Fig. 6. Dependence graph on Type-Level

In both views strong components are illustrated by black bordered colored planes. For a more clear view the strong components were highlighted.

## 4 Conclusions and Future Work

This paper presents an approach to support agile software development and especially the refactoring by the use of software measurement.

The existing Implementation enhanced the Eclipse IDE to support agile software development. Besides Eclipse, there are some other widespread IDE's. On the basis of our reference implementation one can simply adapt the tool to other IDE's.

At the moment the tool supports only Java as a programming language. The enhancement to other programming languages can increase practical benefit.

## References

- [Basili1984] Basili, V., Weiss, D.: A Methodology for Collecting Valid Software Engineering Data. IEEE Transaction on Software Engineering 10, 728–738 (1984)
- [Beck 1999] Beck, K.: Extreme Programming Explained: Embrace Change. Addison-Wesley (1999)
- [Bloch2001] Bloch, J.: Effective Java: Programming Language Guide. Addison-Wesley (2001)
- [Chidamber1993] Chidamber, S., Kemerer, C.: A Metrics Suite for Object Oriented Design. M.I.T. Sloan School of Management (1993)
- [Eclipse] Eclipse Foundation (March 2007), <http://www.eclipse.org>
- [Fowler2000] Fowler, M.: Refactoring: Improving the Design of Existing Code. Addison-Wesley (2000)
- [Henderson1996] Henderson-Sellers, Brian: Object-oriented Metrics: Measures of Complexity. Prentice Hall (1996)
- [ISO/IEC2001] ISO 9126-1: 2001, Software engineering –Product quality – Part 1: Quality model, International Organization for Standardization, Geneva (2001)
- [Kersten2007] Kersten, M.: Unterstützung der agilen Softwareentwicklung durch geeignete Softwaremetriken" Masterthesis, University of Magdeburg (2007)
- [McConnell2004] McConnell, S.: Code Complete. 2 edn. Microsoft Press (2004)
- [Wells2007] Wells, D.: (March 2007) <http://www.extremeprogramming.org>

# Assessment of Software Process and Metrics to Support Quantitative Understanding

Ayça Tarhan<sup>1</sup> and Onur Demirors<sup>2</sup>

<sup>1</sup> Computer Engineering Department, Hacettepe University,  
Beytepe Kampusu, 06532 Ankara, Turkey  
atarhan@hacettepe.edu.tr

<sup>2</sup> Informatics Institute, Middle East Technical University,  
MM Kat:4, 06531 Ankara, Turkey  
demirors@ii.metu.edu.tr

**Abstract.** The use of process metrics and data for quantitative understanding is not very straightforward. If we have an identification of process components and follow a measurement process, we are likely to use process metrics and data effectively. But if we don't have these practices, we can hardly trust on process metrics and data for quantitative understanding. In this paper, we summarize eight case studies that we performed in different industrial contexts. The case studies rely on an assessment approach that investigates suitability of a software process and metrics for quantitative analyses. The approach investigates a process's inner attributes and outer factors as well as a number of usability characteristics for process metrics. We validated the results of the studies via SPC tools. This paper briefs the approach, explains contexts and findings of the case studies, and discusses overall case study results.

**Keywords:** Software measurement, quantitative management, statistical process control.

## 1 Introduction

In any engineering discipline, we measure to understand, control and improve; and software engineering is not an exception. We can use measurement data for understanding a product's quality, for controlling a project's progress, or for improving a process's performance. In any case, we first have to understand quantitatively the current status of what we are dealing with before attempting to manage it. Without quantitative understanding, neither effective control nor improvement is possible.

The use of process metrics and data for quantitative understanding is not very straightforward. We should identify process components which are subject to measurement [9][11], understand the context of measurements made [13][24], and ensure that the metrics we choose and related data satisfy certain requirements for quantitative analyses [2][8]. If we can identify process components and follow a measurement process (as suggested by the most popular process improvement models like CMMI [5] and ISO/IEC 15504 [12]), we are likely to use process metrics and data for quantitative management. But if we don't have these practices, we can hardly trust on process metrics and data for

quantitative analyses. While a number of researchers provide conceptual guidelines to utilize quantitative techniques to understand software processes [3][9][11][17][18][19][23], existing implementations frequently focus on the benefits of quantitative results [1][6][10][14][15][22]. We lack satisfactory guidelines to implement quantitative techniques, especially in emergent organizations [4][7].

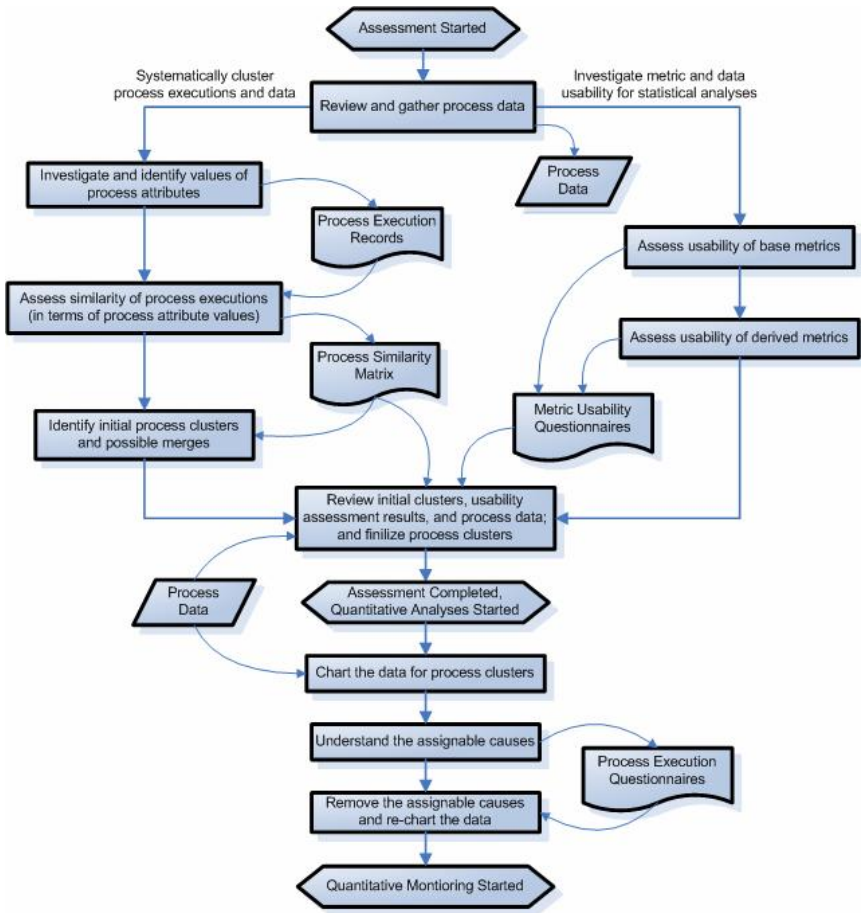
The need for such guidelines encouraged us to carry out a sequence of studies to investigate the suitability of software process and metrics for quantitative understanding. We applied Statistical Process Control (SPC) tools as quantitative techniques. We first utilized control charts in an emergent software organization [6] and derived guidelines [7]. Then we developed an assessment model to guide quantitative analyses [20] and applied it in an emergent organization [21]. By using the model, we investigated a process's inner attributes and outer factors as well as a number of usability characteristics for process metrics. The results of these studies were promising; however, without a proper tool to support the assessment and analyses, it was not very easy for individuals to utilize quantitative techniques. Capturing data and relating it with process executions, identifying homogeneous process subgroups, evaluating usability of process metrics, and performing the analysis and reporting the results all required expertise and careful, dedicated effort. We therefore developed a tool to ease and enhance these activities [16]. The model, the tool, and the process that we defined to perform the assessment constituted our assessment approach.

In this paper, we summarize eight case studies performed in several industrial contexts in which we utilized the assessment approach to support quantitative understanding. Accordingly, section 2 provides a briefing of the approach. Section 3 summarizes the contexts of the case studies. Section 4 discusses case study results, and section 5 states our conclusions and future work.

## 2 Assessment Approach

The assessment approach includes an assessment process that guides the evaluation, an assessment model that defines assets to evaluate a process and metrics, and an assessment tool that supports this evaluation.

The assessment process that we followed during case studies is given in Figure 1. The first step of the assessment process is reviewing and gathering process data. The flow at the left side of the figure is for performing systematic clustering of process executions and data, and the flow at the right side is for investigating metric and data usability for statistical analyses. After we identify initial process clusters and evaluate usability of process metrics, we finalize process clusters and metrics. This is the point where the assessment completes and statistical analyses begin. Control charts, histograms, bar charts, and pareto charts are among SPC tools that we utilized during case studies. We separately put the data for process cluster-process metric pairs on control charts and observe the out-of-control points. We then answer process execution questionnaire for each out-of-control point to understand the assignable causes, if any. We remove data points regarding the assignable causes at each chart and re-chart the data. If the data is under control, then quantitative monitoring is started.



**Fig. 1.** Assessment process for investigating suitability of a software process and metrics for quantitative analyses

The assessment model [20] aims to test the suitability of a software process and metrics for quantitative analyses. It addresses two issues: 1) Systematic clustering of process executions and data, and 2) Investigating metric and data usability for statistical analyses. While working retrospectively, the approach enables the assessment of past process executions and data to identify homogeneous process subgroups and to select metrics that are suitable for SPC implementation.

The first issue is the systematic clustering of process executions and data. Software process data often represent multiple sources that need to be treated separately, and discovering multiple sources requires the careful investigation of process executions. We developed a method for stratification based on a number of process attributes. Stratification is a technique used to analyze or divide a universe of data into homogeneous groups [24], and we want to cluster process executions as stemming from a



single and constant system of chance causes. Our clustering method operates according to the changes in the values of process attributes such as inputs, outputs, activities, roles, and tools and techniques. If the executions of a process show similarity in terms of these attributes, then we assume that process executions form a homogeneous subgroup (or “cluster” as we call it) which consistently performs among its executions. The process cluster, then, is a candidate for statistical control.

The second issue is investigating metric and data usability for statistical analyses. This includes the elaboration of basic measurement practices as well as metric data existence and characteristics. Even if there is not a measurement process defined, we can investigate the practices applied during data collection and analyses. The model identifies a number of metric usability attributes such as metric identity, data existence, data verifiability, data dependability, data normalizability, and data integrability. Metric identity deals with basic characteristics of a metric such as entity and attribute to measure; scale type, unit, formula; and data type and range. Data existence addresses the organization and availability of metric data points. Data verifiability is related with the consistency in metric data recording and storage among executions. Data dependability requires all metric data be recorded as close to its source with accuracy and precision. The awareness of data collectors on metric data (why it is collected, how it is utilized, etc.) plays a significant role in data dependability. Data normalizability and data integrability are related to the usefulness of a metric and should be satisfied if we expect SPC analysis to provide more insight into process understanding and improvement. The model defines questionnaires based on these attributes, and recommends investigating a metric’s usability for statistical analyses prior to implementation. Each usability attribute is rated in four scales: Fully usable (F), largely usable (L), partially usable (P), and not usable (N). Overall usability of a metric is decided based on the ratings of these attributes.

The assets of the assessment model are briefly described below:

*Process Execution Record:* This is a form used to capture the instant values of internal process attributes for a process execution. Actual values of inputs, outputs, activities, roles, and tools and techniques for a specific process execution are recorded on the form. Recorded values are used to identify the merged list of process attribute values which are shown on Process Similarity Matrix.

*Process Similarity Matrix:* This is a form used to verify the values of internal process attributes against process executions. The matrix is created based on the values previously entered into Process Execution Records. The values of internal process attributes are shown in the rows, and process execution numbers are shown in the columns of the matrix. By going over the executions, the values of internal process attributes are questioned and marked if applicable for each process execution. The completed matrix helps us to see the differences among process executions in terms of process attribute values and enables us to identify the clusters of process executions accordingly.

*Process Execution Questionnaire:* This is a form used to capture the assignable causes for a process execution in terms of outer factors such as changes in process performers, process environments, and etc. While working retrospectively on existing process data, a questionnaire is completed for every out-of-control point on a control-chart and answers are used to understand the assignable causes.

*Metric Usability Questionnaire:* This is a form used to investigate the usability of a process metric for quantitative analyses in terms of metric usability attributes. The form includes a number of questions and rules for rating the usability attributes. The questionnaire has two types, for base and derived metrics separately.

The assessment tool [16] has facilities to capture data from outer environment, assess the suitability of a software process and metrics for quantitative analyses, and analyze a software process with respect to its qualifying metrics using SPC techniques like control charts, histograms, bar charts, and pareto charts. Wheeler's tests [24] are used for detecting the assignable causes in a control chart and it is possible to choose the tests to apply. A metric value which is detected as an out-of-control point according to the tests applied can be excluded from the analysis via the tool. The tool supports what-if analysis by merging and splitting identified process clusters. Assessment and analysis results can also be reported and printed by using the tool.

### 3 Case Studies

We performed eight case studies in several industrial contexts in which we utilized the assessment approach to support quantitative understanding. The characteristics of the case studies including contextual description, purpose of implementation, selected processes and metrics, and effort spent are summarized in Table 1.

We worked retrospectively on past process executions and data in all case studies. Our unit of analysis during the studies was "process-metrics" pairs. The assessments were performed by individuals who are software experts. Process performers were the basic information source while trying to capture contextual information of past process executions. The assessment effort was considerably small when compared to process performance effort. For example we spent 14 man-days for performing the assessment and conducting the analysis for the review process in the first case study, and process performers spent 234 man-days for executing 200 reviews. The assessment effort corresponds to %6 of overall process execution effort. We should note that the last three case studies in Table 1 were performed by using the assessment tool, leading considerable effort saving.

We performed process similarity assessment on process executions of each process shown in Table 1, and evaluated usability of selected metrics of each process for quantitative analyses. As a result of process similarity assessment, we identified a number of process clusters whose elements (process executions) were the most similar to each other in terms of the values of internal process attributes. As a result of metric usability assessment, we determined usability of each process metric for quantitative analyses. Table 1 shows the derived metrics that were assessed for quantitative analyses. Base metrics that made up these derived metrics were also assessed for usability prior to assessment of the derived metrics. After the assessment, we conducted quantitative analyses for qualifying metrics of a process, and refined our understanding of the process based on analyses findings. We also included unqualified metrics in the analyses in order to verify the results of our assessments.

**Table 1.** Characteristics of the case studies

Context	Purpose	Process	Metrics	Effort
A - System/software development organization (active for 15 years, has ISO 9001 and AQAP-150 certificates, has 45 development staff)	To understand organizational review process performance	<b>1) Review:</b> 200 process executions, process defined	Non-conformance detection efficiency, non-conformance resolution efficiency, review open period, review open period with respect to non-conformances	14 man-days
B - Project office of a government research agency (develops systems software for 7 years, has ISO 9001 certificate, has 18 development staff)	To understand task management process performance within a single project	<b>2) Task Management (of a military project):</b> 92 process executions, process not defined but executed on a tool	Task effort estimation capability, task effort variance	6 man-days
C - An avionics project of a system/software development organization (the organization is active for 16 years; has ISO 9001, AQAP-150 and CMM L3 certificates; has 75 development staff)	To understand the effects of test design on test development quality	<b>3) Test Design:</b> 48 process executions, process not defined	Test design productivity, percent of test design internal review effort	5 man-days
	To understand the relationship between productivity and quality assurance activities during test development	<b>4) Test Script Development:</b> 48 process executions, process not defined	Test procedure development productivity, percent of test procedure development internal review effort	
		<b>5) Test Development Peer Review:</b> 31 process executions, peer review process defined	Action item density, action item detection efficiency, action item resolution efficiency	
D - A project-based organization (develops software for tele-communication technologies for 6 years, has CMMI L3 certificate, has 80 development staff)	To understand bug-fixing process performance within Project-X	<b>6) Bug Fixing (of Project-X):</b> 42 process executions, process not defined but executed on a tool	Bug aging, bug fixing effort	5,5 man-hours
	To understand recruitment process performance	<b>7) Recruitment:</b> 25 process executions, process not defined	Actual procurement time, procurement time variance	5 man-hours
	To understand bug-fixing process performance within Project-Y	<b>8) Bug Fixing (of Project-Y):</b> 62 process executions, process not defined but executed on a tool	Bug aging, bug fixing effort	5 man-hours

The derived metrics given in Table 1 and the base metrics that made up the derived metrics for each process are explained in Table 2.

**Table 2.** The base and derived metrics utilized during the case studies

Process	Metric	Metric Type	Explanation
Review	Opening date	Base	Review start date
	Closure date	Base	Review finish date
	Number of detected nonconformances	Base	Number of nonconformances detected in the review
	Number of accepted nonconformances	Base	Number of nonconformances accepted in the review
	Review effort	Base	Total effort spent for the review
	Nonconformance resolution effort	Base	Total effort spent for resolving the nonconformances accepted in the review
	Open period	Derived	Closure date – Opening date
	Open period with respect to nonconformances	Derived	Open period / Number of accepted nonconformances
	Nonconformance detection efficiency	Derived	Number of accepted nonconformances / Review effort
	Nonconformance resolution efficiency	Derived	Number of accepted nonconformances / Nonconformance resolution effort
Task Management	Estimated start date	Base	The date estimated to start the task
	Estimated finish date	Base	The date estimated to finish the task
	Actual start date	Base	The date the task was actually started
	Actual finish date	Base	The date the task was actually finished
	Estimated effort	Derived	Estimated finish date - Estimated start date
	Actual effort	Derived	Actual finish date - Actual start date
	Effort estimation capability	Derived	Estimated effort / Actual effort
Test Design	Effort variance	Derived	Estimated effort - Actual effort
	Test design internal review effort	Base	Effort spent for internal reviews performed within the team during test design
	Actual test design effort	Base	Total actual effort spent for test design
	Number of test cases	Base	Total number of test cases designed
	Percent of test design internal review effort	Derived	Test design internal review effort / Actual test design effort
Test Script Development	Test design productivity	Derived	Number of test cases / Actual test design effort
	Test script development internal review effort	Base	Effort spent for internal reviews performed within the team during test script development
	Actual test script development effort	Base	Total actual effort spent for test script development
	Number of test cases	Base	Total number of test cases for which test scripts were developed
	Percent of test script development internal review effort	Derived	Test script development internal review effort / Actual test script development effort
	Test script development productivity	Derived	Number of test cases / Actual test script development effort

**Table 2.** (Continued)

Process	Metric	Metric Type	Explanation
Test Development Peer Review	Test development peer review effort	Base	Effort spent for peer reviews performed by QA staff during test development
	Number of action items	Base	Number of items identified for action (resolution) in test development peer review
	Number of test cases	Base	Total number of test cases for which test development peer review was performed
	Test development peer review update effort	Base	Total effort spent for resolving the issues identified in test development peer review
	Action item detection efficiency	Derived	Number of action items / Test development peer review effort
	Action item density	Derived	Number of action items / Number of test cases
	Action item resolution efficiency	Derived	Number of action items / Test development peer review update effort
Bug Fixing	Creation date	Base	The date bug was reported
	Resolution date	Base	The date bug was resolved
	Bug fixing effort	Base	Effort spent to fix the bug
	Bug aging	Derived	Resolution date - Creation date + 1
Recruitment	Go date	Base	Start of recruitment process
	Due date	Base	Planned start date for new project member
	Start date	Base	Start date for joining of new project member
	Planned procurement time	Derived	Due date - Go date + 1
	Actual procurement time	Derived	Start date - Go date + 1
	Procurement time variance	Derived	Actual procurement time - Planned procurement time

In the next section, we discuss our general observations and findings from case study implementations by providing relevant examples from the cases.

## 4 Discussion

Using the assessment approach during eight case studies helped in many ways and brought some restrictions as we elaborate below.

While identifying process clusters, process performers had a broad understanding of the process and its internal attributes during process executions. This understanding was crucial in interpreting data analysis results and could be attained without the requirement of an explicitly defined process. If there was evidence related to process executions, either on a paper or on a tool, it might be possible to interpret and use process data for quantitative understanding. For example, the second case study of task management process was carried out in a project office of a government research agency. The project team was leaving records on engineering tools for some of their processes. Though there was no definition of task management process, we could analyze estimation capability and effort variance metrics to understand task management process performance.

While identifying metrics to use for quantitative analyses, process performers realized the characteristics of metric data and investigated the existence of basic measurement practices. The investigation was made by answering a questionnaire for the metric under study, and enabled elaboration of metric identity. Though metrics had no prior definitions during the cases, answering the questionnaire captured basic characteristics of each metric and the status of basic measurement practices. For example, the case studies one through five were performed in contexts (organizations A, B, and C) that had no explicit measurement process definition, and we could perform quantitative analyses for process metrics because their data were in fact collected by following basic measurement practices.

We observed that the existence of process definitions could make the assessment easier, although not essential. In some cases having a process definition might even be misleading, since the definition alone did not guarantee consistent execution of the process. For example in the first case study, review process was defined, and we identified two different process clusters in execution: Reviews in which product updates were performed after review, and reviews in which product updates were performed within review. If we did not check every process execution one by one, we would assume a single process cluster addressing all the reviews, and data analyses results would not make much sense due to a mixture of multiple cause systems.

We should note that using the assessment tool made the use of assessment assets easier and speeded up the assessments. Drawing control charts, histograms, pareto charts and bar charts without needing a third party tool was beneficial. The tool reduced the time required for quantitative analyses (for the last three case studies as given in Table 1) by being a central place for collecting, organizing, assessing, and analyzing metric data.

The approach helped to set and refine the understanding of the issues under study (project/process performance, product quality, etc.) in all cases. We clearly observed that the acts of measuring and analyzing a process were themselves means of understanding and improvement. While trying to chart data and interpret chart results, we (together with process performers) checked and refined our understanding of process executions and their contextual information. Trying to identify and eliminate the assignable causes enabled detailed study of individual process executions. Refining process understanding naturally brought recommendations for improvement. For example; in the first case study, review process owner updated the review record to keep the size of each product under review; and in the second case study, the team leader included reviews of task management data in regular progress monitoring.

During the case studies, metric usability evaluation enabled us to select metrics that would succeed in quantitative analyses. This might be important for software organizations that were unfamiliar to, but had bunches of collected data that could in fact be used for quantitative analyses. This was typical for case studies 3, 4, and 5 (in context C) for example. They had bunches of data on papers and tools, but they were using these data basically for project management. The assessment and analyses that we performed on test design, test development and test development peer review processes showed that; test design had an important role in test development quality, and percent of internal review effort spent during test design and development clearly reduced number of issues identified during peer reviews. The quantitative analyses enabled us to see the relationships of test development practices within a single

project, and provided motivation for the institutionalization of test development practices. We think that our approach may provide practicality and flexibility especially for emergent organizations that may not have an established process structure but are willing to understand the performance of their processes in their own contexts based on quantitative data.

We observed that evaluating usability of metrics was supporting but not enough to effectively select the metrics to use for quantitative analyses. Project context and dynamics in which the process was executed (such as project organization, schedule, development life cycle, maturity of development practices, and etc.) were needed to be considered while selecting the metrics. For example, re-charted data of review process in the first case study showed that the company could not use control charts for open period with respect to nonconformances metric confidently, although the metric was evaluated as usable. After the interviews we could detect that the schedule of the projects played a significant role in the open periods of review records. Elaboration on process metrics prior to analyses requires special attention from this perspective.

By process similarity assessment we could systematically identify process clusters, which is difficult to achieve especially for software processes that require considering human-related and environmental issues. We observed that the identification of process clusters was closely related to the purpose of quantitative analyses. For example, in case study one, the purpose was to understand organizational review process performance and we identified clusters in such a way that the variation in review practices would be at a minimum. In case study two, on the other hand, the purpose was to understand task management process performance within a single project so that we identified process clusters with respect to the types of tasks that are being managed.

We had a number of constraints related to the assessment model and its applications. The first one was retrospective characteristic of the case studies. We questioned the attributes of past executions and since we worked on what was available, we had some difficulties in catching implementation details. Organizing a prospective case study would support better understanding of process executions. Second, we performed the assessments by ourselves and by consulting process performers as required for the case studies one through five. This was partly due to the unavailability of a practical tool. After developing the tool, process performers conducted the assessments by themselves for the last three case studies and suggested a number of improvements for usability. For example, the use of metric usability questionnaire had subjectivity in some parts and required expertise in measurement and analysis concepts. We need to better verify whether the model and the tool is easily usable by the company staff. The third constraint was that metric usability evaluation provided information on a metric's usability for statistical analysis, but it did not state whether the utilization would be effective. The selection of effective metrics for a process needs further elaboration. Fourth, we could not generalize the results from our case studies since the variety in the type of organizations was limited. The usability of the model needs to be tested by conducting more case studies in various contexts.

## 5 Conclusion

Quantitative techniques are crucial for process understanding and improvement, but encouraged in industry mostly by adopting organizational maturity or process capability frameworks offered by process improvement models. These models demand investment of time, effort, and money for several years, which are difficult to afford. More practical methods are needed, especially for emergent organizations, to understand and manage processes based on quantitative data. To address this difficulty, we developed an assessment approach to test the suitability of a software process and metrics for quantitative analyses.

The approach includes an assessment process that guides evaluations, an assessment model that defines assets to carry out evaluations, and an assessment tool to support these evaluations. The approach helps in understanding the context in which process data are generated, in identifying homogeneous subgroups of process executions, and in selecting metrics to use for quantitative analyses.

We applied the assessment approach for eight processes and related metrics in several organizational contexts. The results of quantitative analyses were consistent with the findings of our assessments. In other words, the process clusters that we identified as the result of process similarity assessment were detected as under control on control-charts with respect to process metrics that qualified usability assessment. The results of applications showed that the approach increased the level of confidence in quantitative analyses and enabled process understanding based on quantitative data.

We have recently initiated a prospective case study on system test process of an organization which is on the way to get CMMI L3. We expect prospective studies will better capture contextual information of process executions and data. As future work, we target more case studies using the assessment approach, especially in emergent contexts. This will provide better insight for general usability of our approach.

## References

1. Barnard, J., Carleton, A.: Analyzing a Mature Software Inspection Process Using Statistical Process Control. In: National SEPG Conference, Pittsburgh (1999)
2. Burr, A., Owen, M.: Statistical Methods for Software Quality. Thomson Publishing Company (1996) ISBN: 1-85032-171-X
3. Card, D.: Statistical Process Control for Software? IEEE Software, 95–97 (May 1994)
4. CMU/SEI: Process Maturity Profile of the Software Community – 2000 Year End Update. Presentation (March 2001)
5. CMU/SEI, CMMI Product Team: CMMI for Development. CMMI-SE/SW V1.2, CMU/SEI-2006-TR-008 (August 2006)
6. Demirors, O., Sargut, K.U.: Utilization of a Defect Density Metric for SPC Analysis. In: 13th International Conference on Software Quality, Dallas, Texas (October 2003)
7. Demirors, O., Sargut, K.U.: Utilization of Statistical Process Control (SPC) in Emergent Software Organizations: Pitfalls and Suggestions. Software Quality Journal 14(2), 135–157 (2006)
8. Fenton, N.E., Pfleeger, S.L.: Software Metrics: A Rigorous and Practical Approach, 2nd edn. PWS Publishing Company (1997)



9. Florac, A.W., Carleton, A.D.: *Measuring the Software Process: Statistical Process Control for Software Process Improvement*. Pearson Education (1999) ISBN 0-201-60444-2
10. Florac, A.W., Carleton, A.D.: *Statistical Process Control: Analyzing a Space Shuttle On-board Software Process*. IEEE Software, 97–106 (July/August, 2000)
11. Humphrey, W.: *Managing the Software Process*. Addison-Wesley Publishing Company (1989) ISBN 0-201-18095-2
12. ISO/IEC: ISO/IEC 15504: *Information Technology – Process Assessment* (2003-2006)
13. ISO/IEC: ISO/IEC 15939: *Software Measurement Process* (2002)
14. Jacob, L., Pillai, S.K.: *Statistical Process Control to Improve Coding and Code Review*. IEEE Software, 50–55 (May/June 2003)
15. Jalote, P., Dinesh, K., Raghavan, S., Bhashyam, M.R., Ramakrishnan, M.: *Quantitative Quality Management through Defect Prediction and Statistical Process Control*. In: *Proceedings of Second World Quality Congress for Software* (September 2000)
16. Kirbas, S., Tarhan, A., Demirsors, O.: *An Assessment and Analysis Tool for Statistical Process Control of Software Processes*. In: *SPICE Conference, Seoul, May 9-11 (2007)*
17. Lantzy, M.A.: *Application of Statistical Process Control to Software Processes*. In: *Proceedings of the Ninth Washington Ada Symposium on Empowering Software Users and Developers*, pp. 113–123 (1992)
18. Paulk, M.C.: *Practices for High Maturity Organizations*. In: *Proceedings of the 1999 Software Engineering Process Group Conference, Atlanta, Georgia, March 1999*, pp. 28–31 (1999)
19. Radice, R.: *Statistical Process Control for Software Projects*. In: *10th Software Engineering Process Group Conference, Chicago, Illinois (March 1998)*
20. Tarhan, A., Demirsors, O.: *Investigating Suitability of Software Process and Metrics for Statistical Process Control*. In: Richardson, I., Runeson, P., Messnarz, R. (eds.) *EuroSPI 2006*. LNCS, vol. 4257, pp. 87–98. Springer, Heidelberg (2006)
21. Tarhan, A., Demirsors, O.: *Remarks from SPC Trial for an Emergent Organization*. Presentation. In: *European SEPG Conference, Amsterdam, June 12-15 (2006)*
22. Weller, E.: *Practical Applications of Statistical Process Control*. IEEE Software, 48–55 (May/June 2000)
23. Wheeler, D.J.: *Understanding Variation: The Key to Managing Chaos*. SPC Press, Knoxville (1993)
24. Wheeler, D.J.: *Advanced Topics in Statistical Process Control*. SPC Press, Knoxville (1995)

# Developing and Applying a Consolidated Evaluation Framework to Analyze Test Process Improvement Approaches

Ayaz Farooq and Reiner R. Dumke

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

**Abstract.** Following the general software process improvement initiatives, improvement models particular to the software test process have also been developed. Although these models have been based on nearly similar principles, yet they differ in a variety of aspects. To compare and critically analyze strengths and weaknesses of these test process models, we need a common analysis framework. Existing SPI evaluation frameworks focus only on some basic and key process characteristics and elements. A number of critical success factors for software process improvement have been identified in literature which impose higher level of requirements on SPI approaches. We present a consolidated evaluation framework derived from critical success factors/literature review and apply it to analyze well known test process improvement approaches.

**Keywords:** Software process improvement, software test process, test process improvement, Testing Maturity Model, Critical success factors.

## 1 Introduction

Software process improvement (SPI) is a systematic procedure to improve the performance of an existing process system by changing current processes or updating new processes in order to correct or avoid problems identified in the old process by means of a process assessment [1]. Many different kinds of process improvement models, and approaches [1, Ch.5-9][2, Ch.2][3, p.207-222] sprang out in recent years and are being practised by industry today.

Software test process, although embedded into the overall development process, comprises distinctly recognizable set of activities within the software process. A major portion of software project resources is consumed by activities of the testing process. Therefore, improvement of the testing process is deemed to provide overall quality gains both to the software process and product.

A number of competing test process improvement approaches exist today such as Testing Maturity Model (TMM) [4], Test Process Improvement (TPI) model <sup>1</sup>, Metrics based Verification and Validation Maturity Model ( $MB - V^2M^2$ ) [5],

---

<sup>1</sup> <http://www.sogeti.nl/Home/Expertise/Testen/TPI.jsp>

and Test Improvement (TIM) model [6] etc. Despite sharing similar founding principles behind these models, a variety of differences from various aspects may also be observed. Raw comparisons of some of these approaches have already appeared in [7][8][9]. However, in order to better understand and evaluate these approaches a systematic comparison methodology is missing in literature.

Although a few evaluation techniques for (general) software process improvement models exist such as [10][11], but these techniques compare only some fundamental SPI model characteristics. A step further, we have developed our evaluation approach at a higher level of abstraction by deriving analysis criteria from reported critical success factors (CSFs) [2, Ch.3][12][13] for SPI. Our proposed evaluation framework, although applicable equally to both software process and test process improvement models, is being applied here to evaluate available test process improvement approaches only.

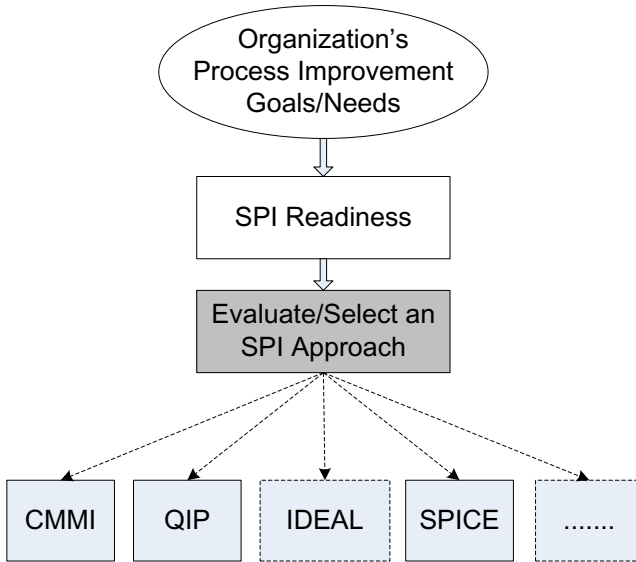
The rest of the paper is organized as follows. An outline of steps to implementing an SPI methodology is discussed in the following section. Section 3 existing SPI evaluation frameworks. Critical success factors (CSFs) for SPI are summarized in section 4. Evaluation framework proposed by this paper is contained in section 5. Section 6 reviews and analyzes available software test process improvement approaches using our framework. Finally, the paper concludes in section 7.

## 2 Implementing an SPI Approach

Within the context of adopting a software process improvement methodology, two aspects are involved. On one hand are a software organization's process improvement needs and goals. On the other hand are several available process improvement models and techniques. What connects both of these ends is the decision making process to initiate an SPI program. This process involves careful consideration. Figure 1 outlines these steps.

The first step in this decision making process is the determination of an organization's readiness to undertake a software process improvement program. In this connection, Wilson et. al [14] suggested a framework to support companies to evaluate necessary organizational requirements for establishing SPI programs. This framework consists of several questions grouped into different perspectives and provides guidelines for interpreting these questions to determine an organization's preparedness to implement an SPI program.

The next step is to choose and adopt or adapt a suitable process improvement model (appropriate to the organization's specific constraints and needs) out of the many competing models and approaches available. The names of several SPI and test process improvement models have already been mentioned in section 1. At times, most of these models may seem quite similar but they may differ as well from various aspects. Availability of a systematic comparative analysis of candidate process improvement approaches could be helpful to make a judicious choice. Halvorsen and Conradi [11] summarized different approaches that can



**Fig. 1.** SPI Selection/Implementation Strategy

help to cope with this issue. They mention four main classes of existing comparison methods for SPI frameworks.

1. Characteristics Comparison
2. Framework Mapping
3. Bilateral Comparison
4. Needs Mapping

Characteristics comparison method probably is the most commonly used. It involves comparing SPI frameworks based on their general attributes, features and areas of interest etc. The next section explains available SPI comparison frameworks in more details and discusses their limitations.

### 3 Existing SPI Evaluation Methodologies

We view the assessment of the process improvement models in two ways, a *post-evaluation* and a *pre-evaluation*. Post-evaluation is dynamic in nature which involves application of a process improvement model and then analyzing & studying positive/negative results and gains achieved. Accounts of case studies, lessons learnt and experience reports are examples of this kind of evaluation. This paper is not concerned with such analysis.

Conversely, pre-evaluation is a kind of static technique where we examine characteristics and elements of a given SPI methodology itself to see which issues

are addressed by it, which improvement areas have been focused on, and which type of guidelines are provided etc.

- Saiedian and Chennupati [10] presented an evaluative framework for SPI models which can be used to make comparative analysis for selecting a process improvement model. The components of their framework focus on *what*, *which*, *how*, and *where* aspects of a process improvement model.
- Halvorsen and Conradi [11] presented a taxonomy to compare SPI models. They divided the characteristics of a process model into five categories, *general*, *process*, *organization*, *quality*, and *result*.
- Recently, as part of his PhD dissertation, Komi-Sirviö [2, Ch.3] developed another evaluation criteria based on some collected critical success factors and lessons learnt from case studies and analyzed a few SPI models based on this criteria.
- A comparative analysis of test process improvement models was performed by Swinkels [8]. He used Testing Maturity Model (TMM) as a reference model and compared other related models with it on the basis of some arbitrarily selected model characteristics. Although the conclusions were useful in highlighting strengths and weaknesses of the observed models, yet it was only a bilateral kind of comparison among a within a certain context.
- Farooq and Dumke [7] critically analyzed structure and content of the Testing Maturity Model (TMM), highlighted some of its weaknesses and suggested few improvements.

A limitation of the above mentioned techniques is that their evaluation criteria are based upon some very basic model elements and characteristics of SPI models. A finer analysis based upon most effective success criterion is not possible with these available approaches. Furthermore, these are somewhat detached techniques which complement each other. A consolidated approach with inclusion of most recently suggested SPI model characteristics could improve existing drawbacks.

## 4 Critical Success Factors for Software Process Improvement

With so many SPI models and methodologies the question arises how can these approaches themselves be improved? What factors have a positive effect on the success of these improvement activities? What characteristics they must contain? Although the SPI literature abounds with case studies of successful companies and accounts of their SPI programs, yet we need concrete answers to these questions.

Determination of critical success factors (CSF) for a given SPI technique can effectively answer these types of questions. Several sets of CSFs for software process improvement have been suggested, summarized, and empirically tested by many researchers. Dyba [12] has explored and validated a number of factors affecting the success of an SPI program. These are:

1. Business orientation
2. Involved leadership
3. Employee participation
4. Concern for measurement
5. Exploitation of existing knowledge
6. Exploration of new knowledge

Niazi et. al [13] collected several CSFs from literature and have identified 7 key CSFs through an empirical study. These include:

1. Higher management support
2. Training
3. Awareness
4. Allocation of resources
5. Staff involvement
6. Experienced staff
7. Defined SPI implementation strategy

Conradi and Fuggetta [15] have also highlighted some key aspects which can be effective in improving a process improvement program. After a review of many industrial experiences, surveys and argumentations for SPI issues, Komi-Sirviö [2, Ch.3] summarized factors for a successful SPI initiative into several main classes and sub-classes as given below.

1. Improvement management
  - (a) General guidance
  - (b) Staffing the SPI initiative
  - (c) Training
2. Commitment
3. Cultural issues
4. Plan
  - (a) Current state analysis
  - (b) Goal definition
  - (c) Improvement planning
5. Do
6. Check
7. Act

It is evident that many of these CSFs are quite similar and rather overlap in some cases. The concept of critical success factors may also be used to compare, contrast, and evaluate software process improvement approaches. In this case, it will serve as a kind of characteristics comparison method discussed in the previous section. Process improvement models may be analyzed whether or not they address issues related to a given CSF. For example, concern for measurement is one of the CSFs for SPI. We may investigate an SPI approach as to what extent does it exploit measurements to track progress of activities and the process. The next section discusses our proposed evaluation framework which has been derived from above mentioned CSFs.

## 5 Consolidated Evaluation Framework

This framework will attempt at the more fine and superior characteristics extracted from most effective CSFs and contemporary SPI model requirements. Our evaluation framework consists of two components,

- a) desired SPI model/method characteristics
- b) a rating scale describing the level of satisfaction of a given characteristic for the SPI method.

### 5.1 Desired Characteristics

After an extensive review of SPI related literature [1][15][16], validated critical success factors [12][13], recent PhD dissertations related to process improvement [2][3][17], and SPI experience reports [18][19] we organize our evaluation criteria into four categories of characteristics.

C1: Compliance with process standards

C2: Exploitation of measurement

C3: People issues

C3.1: Leadership/employee participation

C3.2: Training

C3.3: Awareness

C4: Process management issues

C4.1: Business orientation

C4.2: Knowledge management and learning

C4.3: Defined implementation methodology

#### C1: Compliance with Process Standards

Software engineering process related standards are aimed at providing universally accepted guidelines for establishing, assessing, and improving software processes. Some well known standards related to software processes are ISO/IEC standards 9000:2000, 9001:2000, 12207 (Standard for Information Technology-Software Life Cycle Processes), 15504 (SPICE), and SWEBOK (Software Engineering Body of Knowledge) of IEEE.

ISO/IEC 15504 (SPICE) consists of several parts and defines detailed requirements for software process assessment (SPA) and software process improvement (SPI) procedures [3, p.65-70]. To evaluate SPI methods, we reproduce following model related evaluation criteria from Braungarten's PhD dissertation [3, p.71].

- The process improvement model must provide a process reference model compliant with the requirements of ISO/IEC Standard 15504.
- The process improvement model's assessment process must observe the regulations of ISO/IEC Standard 15504.
- The process improvement model's process assessment model must represent a mapping of the process reference model and the measurement framework of ISO/IEC Standard 15504.

## C2: Exploitation of Measurement

The role of measurement for process assessment and improvement has been well established [20][21][22]. The most well known process maturity model (CMMI) contains a dedicated measurement & analysis process area [23]. Concern for measurement has been proved to be one of the critical success factors for SPI [12].

Measurement facilitates understanding, controlling, monitoring, predicting, and evaluating software process and making objective decisions during the course of action of process activities. Measurement and analysis of process activities is a helpful tool for process managers who need to constantly track process progress and to identify improvement needs. Borrowing from [12], we define the exploitation of measurement as the extent to which the software organization collects and utilizes quality data to guide and assess the effects of SPI activities. From the perspective of involvement of measurement into the process model, we outline following characteristics:

- The mode of measurement and analysis within the process improvement model is governed by the measurement framework of ISO/IEC Standard 15504.
- The process improvement model contains guidelines on supporting and tracking process activities with measurement-based information collection, analysis, and application.

## C3: People Issues

People are an intellectual capital of an organization [24, Ch.25]. Software process is run by people, higher leadership, process managers, developers, and other kinds of employees. The importance of the role of people in process success is also endorsed by the development of Team Software Process (TSP) and People Capability Maturity Model (People CMM) [24] by Software Engineering Institute (SEI). The skills, capabilities, attitudes, and opinions of people inevitably affect process activities.

Different people issues like their involvement, motivation, support, awareness, training, and experience have been identified as CSFs. The importance of leadership commitment and involvement in the implementation of improvement actions is beyond any doubt [16] while employee participation and the way people are treated has been noted as a crucial factor in organizational management and development [12]. Key to employee participation lies in respecting their esteem, social, safety, and physiological needs [24]. Furthermore, employee training is also essential which develops the skills and knowledge of people so they can perform their roles effectively and efficiently. Such training can be at the organization or project basis. Awareness is another personnel issue. Since SPI is an expensive and long-term approach and it takes a long time to realize the real benefits of this approach, support of management and practitioners and to successfully continue SPI initiatives can only be achieved through sufficient awareness of SPI in organizations [12].

We refer to people issues as a composite aspect involving following process characteristics.



- The process improvement model stresses upon commitment of the leadership and higher management, and application of employee's knowledge and experience towards process improvement activities.
- The process improvement model contains clear guidelines for necessary developer and employee training to appropriately execute process activities.
- The process improvement model promotes, through awareness events, the long-term benefits of SPI among the higher management and the staff members of the organization.

#### **C4: Process Management Issues**

Efficient management of process activities is central to an effective and optimized process. The primary concept behind all kinds of process improvement models and approaches is simply to manage process activities. IEEE SWEBOK [25] gives a broader picture of software process management cycle as consisting of four phases i.e, establishment of process infrastructure, planning, implementation, and evaluation. Process management involves application of knowledge, skills, tools, and techniques to meet process requirements.

The core reason to initiate any process improvement program is attainment of some pending business goals. Therefore, success of any SPI program is heavily dependent upon its correspondence with business needs [16]. Dyba [12] defines business orientation as the extent to which SPI goals and actions are aligned with explicit and implicit business goals and strategies.

Knowledge management is concerned with creating, preserving and applying the knowledge that is available within an organization. Software engineering itself is a knowledge and learning intensive field. Techniques for software process improvement by knowledge management have been presented in [26][27] and a review can be found in [2, p. 56-60]. Knowledge management in SPI involves exploitation of existing knowledge as well as exploration of new knowledge both of which form CSFs for SPI.

Most process improvement models focus on the *what* aspect of the process activities. Identification of what activities to perform in an SPI program is as much necessary as are the details on *how* to implement them. A defined SPI implementation methodology is an empirically verified critical success factor that can play a vital role in the implementation of SPI programs [13][28]. Niazi et. al [29] have also presented a model for the implementation of software process improvement.

Process management is also a kind of composite criteria which we have derived from several CSFs cited in literature. The specific process characteristics relating to process management are:

- The process improvement model requires conformity of process goals and actions with explicit and implicit business goals and strategies.
- The process improvement model provides guidelines on using established ideas, paradigms, technologies, strategies, and knowledge for enhancing process improvement capabilities.
- The process improvement model promotes learning through discovery and experimenting with ideas, paradigms, technologies, strategies, and knowledge.

- The process improvement model mandates establishing and defining an SPI implementation strategy corresponding to the process improvement approach and organizational constraints.

## 5.2 Rating Scale

Although many SPI methods and models are based on nearly similar principles, yet a variability among these models exists over addressing different process improvement aspects/characteristics. A given aspect may be stressed and elaborated well in a given SPI method, may be given some minor consideration, or it may have been completely ignored as well. However, a strictly *yes/no* answer perhaps will not be true representative of the state whether a method can be said to have met a characteristic or not.

Therefore, we propose three rating scales for the purpose.

**N; Not-addressed:** The characteristic is completely ignored and no relevant guidelines exist.

**P; Partially-addressed:** Some insufficient guidelines exist pertaining to the characteristic.

**F; Fully-addressed:** The characteristic is well addressed and detailed relevant guidelines exist.

The three scales mentioned above are opinion-centric, i.e. it involves opinion and personal judgement of the evaluator how he views presence of a characteristic in a given SPI approach. Since these characteristics are qualitative in nature, so a precise and objective evaluation is not possible. This, however, does not heavily diminish meaningfulness of the evaluations which can still be considerably effective.

## 6 Analysis of Existing Test Process Improvement Approaches

As compared to software process improvement research, fewer efforts exist in the direction of test process improvement. These include some maturity, assessment, and improvement models, standards, few mathematical/dynamic models, sets of test process metrics, and a number of individual techniques for improving some aspect of the test process. Obviously, such an evaluation criteria that we proposed is not applicable to all of these test process improvement approaches. In the following we evaluate only a few of these models using our evaluation framework.

### 6.1 Testing Maturity Model (TMM)

Testing Maturity Model (TMM) was developed by Ilene Burnstein [4] to assist and guide organizations focusing on test process assessment and improvement.

It is perhaps the most comprehensive test process assessment and improvement model. This model consists of a set of five maturity levels, a set of maturity goals and subgoals and associated activities, tasks and responsibilities (ATRs), and an assessment model. A detailed criticism of TMM has earlier appeared in [7] which suggests some improvements to model structure, an update to its assessment model, and an expansion of its process areas.

Next we assess TMM with the help of our evaluation framework.

*C1: Compliance with process standards:* As observed above TMM did not particularly follow requirements of ISO/IEC 15504. However, it is heavily influenced by CMM V 1.1 and derives most of its concepts, terminology, and model structure. Furthermore, its assessment model closely follows SPICE's assessment guidelines. Burnstein [4, p. 563-568] also argues that TMM can be used in conjunction with other process improvement standards and models. We conclude that TMM *partially (P)* supports this characteristic.

*C2: Exploitation of measurement:* TMM Level 4 (Management and Measurement) contains Maturity Goal 4.2 (establish a test measurement program) that outlines requirements for a test measurement program. Additionally, Maturity Goal 3.4 (control and monitor the testing process) also advises on the use of test-related measurements to be collected for this purpose. We can assume that C2 is *fully (F)* satisfied by TMM.

*C3: People Issues:* The provision of Maturity Goal 3.1 (establish a test organization) can be considered to *partially (P)* address criteria C3.1 (leadership/employee) participation and C3.3 (awareness). However, Maturity Goal 3.2 (establish a technical training program) contains adequate guidelines on employee training. C3.2 is, therefore, *fully (F)* met.

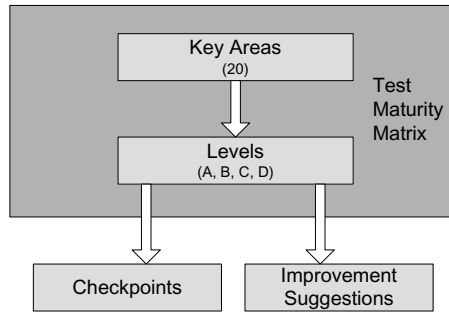
*C4: Process management issues:* The very first Maturity Goal 2.1 (develop testing and debugging goals and policies) is about defining goals according to organizational and project needs. C4.1 is hence *fully (F)* fulfilled. Maturity Goal 5.1 (defect prevention) encourages an organization to formally classify, log, and analyze its defects. Further directions on exploiting knowledge and learning are not given. So, C4.2 is *partially (P)* satisfied. Maturity Goal 2.2 (initiate a test planning process) calls for outlining strategies, developing test design specifications, and test cases. It can be considered to stress upon defining some elements of a process implementation methodology. C4.3 is also only *partially (P)* satisfied by TMM.

## 6.2 Test Process Improvement (TPI) Model

Test Process Improvement (TPI) <sup>2</sup> model is an industrial initiative to provide test process improvement guidelines. The model is composed of a maturity model, test maturity matrix, a checklist, and improvement suggestions. It defines

---

<sup>2</sup> <http://www.sogeti.nl/Home/Expertise/Testen/TPI.jsp>



**Fig. 2.** The TPI Model

3-4 maturity levels and several checkpoints inside these levels, and 20 key areas relating to different test process aspects. Structure of the TPI model is shown in figure 2 below.

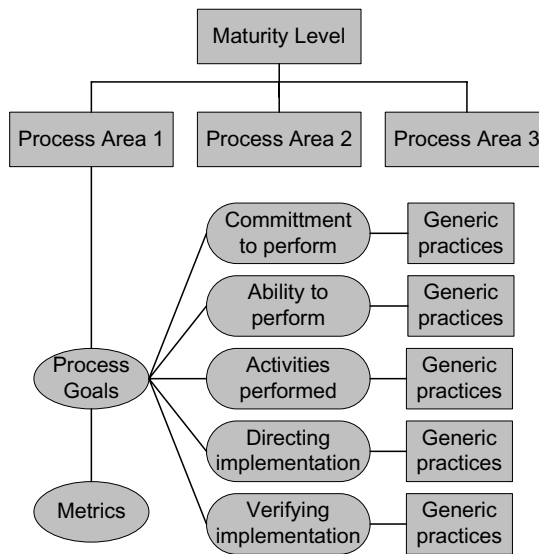
Success of the TPI model has been reported in two worldwide surveys [30][31] conducted by Sogeti (the organization which developed this model). A critical review of this model and comparison to other test process improvement models has already been given by Swinkels [8]. In the following we give our analysis of this model.

*C1: Compliance with process standards:* TPI does not follow an improvement path compatible with ISO/IEC 15504 and its assessment model, too, does not make any reference to the assessment guidelines of ISO/IEC 15504 standard. Criteria C1 is *not (N)* satisfied.

*C2: Exploitation of measurement:* The key area Metrics within this model defines role of measurement. However these guidelines are not governed by measurement framework of ISO/IEC 15504. C2 is also *partially (P)* addressed.

*C3: People Issues:* The key areas Office Environment and Commitment and Motivation concern with involving managers and employees and providing them a suitable working environment. C3.1 is *fully (F)* satisfied and so is C3.2 with the presence of Testing Functions and Training key areas. C3.3 (awareness), however, is not explicitly mentioned in any key area within TPI. C3.3 is *not (N)* met.

*C4: Process management issues:* The key area Test Process Management defines four steps (planning, execution, maintaining and adjusting) for every activity/process to be executed. However, no other key area relates to establishing appropriate process goals based on business objectives. C4.1 is *not (N)* met. The key areas Defect Management and Testware Management pertains to managing relevant/useful process knowledge which can serve as an experience database for future use. C4.2 is *fully (F)* present. Scope of Methodology key area is about establishing an implementation methodology comprising activities, procedures, regulations, and techniques. Therefore, C4.3 is considered *fully (F)* satisfied.



**Fig. 3.** Metrics based V&V Maturity Model

### 6.3 Metrics Based Verification and Validation Maturity Model ( $MB - V^2M^2$ )

( $MB - V^2M^2$ ) [5] is an extension of TMM and also inherits influences from CMM. The model extends its maturity framework to verification & validation in contrast to TMM's focus on testing only. It promises to provide consistent description of model elements, add few process areas, provide a metric base to measure V&V process improvement, and attempt to make some updates to the assessment process. Structure of this model is shown in figure 3.

We have found that except for the process road-map given in [5], any further information on development status of this model is not available in literature. We assume that the model is perhaps still under development or has been abandoned. Therefore, we disregard this model for our evaluation due to lack of sufficient model details and information on its current status.

However, as evident from its structure shown in figure 3, if and when  $MB - V^2M^2$  model is fully developed, it could be a significant improvement in the direction of testing related maturity models. Another such under-development approach is TMMi Foundation <sup>3</sup> which is also being disregarded here due to lack of accurate information on its current development status.

Table 1 summarizes our evaluation findings where letters N, P, and F represent our rating scale as described in section 5.2 above.

<sup>3</sup> <http://www.tmmifoundation.org/>

**Table 1.** Evaluation Summary

Characteristic	TMM	TPI	$MB - V^2M^2$
C1	P	N	-
C2	F	P	-
C3.1	P	F	-
C3 C3.2	F	F	-
C3.3	P	N	-
C4.1	F	N	-
C4 C4.2	P	F	-
C4.3	P	F	-

## 7 Conclusions

After considering to improve its testing process, an organization faces the challenge of choosing a particular improvement model which is compatible with its goals, needs, and other constraints. It must be able to critically analyze available test process improvement approaches. In addition to application of a generic evaluation framework covering broad aspects of process improvement models, a more focused evaluation criteria based on finer-level model requirements can greatly help. Such an evaluation can also reveal improvement suggestions to improve an implemented process improvement model. We have applied our evaluation framework and critically analyzed few test process improvement models. Our evaluation shows that although Testing Maturity Model (TMM) addresses more process improvement aspects in comparison to Test Process Improvement (TPI) model, yet it lacks adequate guidelines on many process improvement issues.

## References

1. Wang, Y., King, G.: Software engineering processes: principles and applications. CRC Press, Inc., Boca Raton (2000)
2. Komi-Sirviö, S.: Development and Evaluation of Software Process Improvement Methods. PhD thesis, Faculty of Science, University of Oulu, Oulu, Finland (2004), <http://www.vtt.fi/inf/pdf/publications/2004/P535.pdf>
3. Braungarten, R.: The SMPI model: A stepwise process model to facilitate software measurement process improvement along the measurement paradigms. PhD thesis, University of Magdeburg, Magdeburg, Germany (2007)
4. Burnstein, I.: Practical Software Testing: A Process-oriented Approach. Springer Inc., New York (2003)
5. Jacobs, J.C., Trienekens, J.J.M.: Towards a metrics based verification and validation maturity model. In: STEP 2002: Proceedings of the 10th International Workshop on Software Technology and Engineering Practice, p. 123. IEEE Computer Society Press, Washington (2002)
6. Ericson, T., Subotic, A., Ursing, S.: TIM a test improvement model. J. Softw. Test., Verif. Reliab. 7(4), 229–246 (1998)

7. Farooq, A., Hegewald, H., Dumke, R.R.: A critical analysis of the Testing Maturity Model. Metrics News, Journal of GI-Interest Group on Software Metrics 12(1), 35–40 (2007)
8. Swinkels, R.: A comparison of TMM and other test process improvement models. Technical report, Frits Philips Institute, Technische Universiteit Eindhoven, Netherlands (2000),  
<http://is.tm.tue.nl/research/v2m2/wp1/12-4-1-FPdef.pdf>
9. Kulkarni, S.: Test process maturity models - yesterday, today and tomorrow. In: Proceedings of the 6th Annual International Software Testing Conference in India (2006)
10. Saiedian, H., Chennupati, K.: Towards an evaluative framework for software process improvement models. J. Syst. Softw. 47(2–3), 139–148 (1999)
11. Halvorsen, C.P., Conradi, R.: A taxonomy to compare SPI frameworks. In: Ambriola, V. (ed.) EWSPT 2001. LNCS, vol. 2077, pp. 217–235. Springer, Heidelberg (2001)
12. Dyba, T.: An empirical investigation of the key factors for success in software process improvement. IEEE Trans. Softw. Eng. 31(5), 410–424 (2005)
13. Niazi, M., Wilson, D., Zowghi, D.: Critical success factors for software process improvement implementation: an empirical study. Software Process: Improvement and Practice 11(2), 193–211 (2006)
14. Wilson, D.N., Hall, T., Baddoo, N.: A framework for evaluation and prediction of software process improvement success. J. Syst. Softw. 59(2), 135–142 (2001)
15. Conradi, R., Fuggetta, A.: Improving software process improvement. IEEE Software 19(4), 92–99 (2002)
16. Humphrey, W.S.: Managing the software process. Addison-Wesley Longman Publishing Co., Inc., Boston (1989)
17. van Solingen, R.: Product Focused Software Process Improvement: SPI in the Embedded Software Domain. PhD thesis, Eindhoven University of Technology, Eindhoven, The Netherlands (2000),  
<http://alexandria.tue.nl/extra2/200000702.pdf>
18. Haug, M., Olsen, E.W., Bergman, L.: Software Process Improvement: Metrics, Measurement and Process Modelling. Software Best Practice 4. Springer Verlag, Berlin (2001)
19. O'Hara, F.: European experiences with software process improvement. In: ICSE 2000: Proceedings of the 22nd international conference on Software engineering, pp. 635–640. ACM Press, New York (2000)
20. Dumke, R.R., Braungarten, R., Blazey, M., Hegewald, H., Reitz, D., Richter, K.: Software process measurement and control - a measurement-based point of view of software processes. Technical report, Dept. of Computer Science, University of Magdeburg (2006)
21. Dumke, R.R., Hegewald, M.B.H., Reitz, D., Richter, K.: Causalities in software process measurement and improvement. In: MENSURA 2006: Proceedings of the International Conference on Software Process and Product Measurement, Cádiz, Spain, Servicio de Publicaciones de la Universidad de Cádiz, pp. 42–52 (2006),  
<http://paginaspersonales.deusto.es/cortazar/doctorado/articulos/2006/MENSURA2006Proceedings.pdf>
22. Pfleeger, S.L., Rombach, H.D.: Measurement based process improvement. IEEE Softw. 11(4), 8–11 (1994)

23. Goldenson, D.R., Jarzombek, J., Rout, T.: Measurement and analysis in capability maturity model integration models and software process improvement. *CrossTalk The Journal of Defense Software Engineering* (2003),  
<http://www.stsc.hill.af.mil/crosstalk/2003/07/goldenson.html>
24. Sommerville, I.: *Software Engineering*, 8th edn. Pearson Education Limited, Harlow (2007)
25. Abran, A., Bourque, P., Dupuis, R., Moore, J.W. (eds.): *Guide to the Software Engineering Body of Knowledge - SWEBOK*. IEEE Press, Piscataway (2004)
26. de Almeida Falbo, R., Borges, L.S.M., Valente, F.F.R.: Using knowledge management to improve software process performance in a CMM level 3 organization. In: *QSIC 2004: Proceedings of the Fourth International Conference on Quality Software*, pp. 162–169. IEEE Computer Society Press, Washington (2004)
27. Kneuper, R.: Supporting software process using knowledge management. *Handbook of Software Engineering and Knowledge Engineering* (2001),  
<ftp://cs.pitt.edu/chang/handbook/37b.pdf>
28. Rainer, A., Hall, T.: Key success factors for implementing software process improvement: a maturity-based analysis. *J. Syst. Softw.* 62(2), 71–84 (2001)
29. Niazi, M., Wilson, D., Zowghi, D.: A model for the implementation of software process improvement: A pilot study. In: *QSIC: Proceedings of the 3rd International Conference on Quality Software*, pp. 196–203. IEEE Computer Society, Washington (2003)
30. Koomen, T.: *Worldwide survey on Test Process Improvement*. Technical report, Sogeti (2002)
31. Koomen, T.: *Worldwide survey on Test Process Improvement*. Technical report, Sogeti (2004)



# Using Controlled Experiments for Validating UML Statechart Diagrams Measures

José A. Cruz-Lemus, Marcela Genero, and Mario Piattini

ALARCOS Research Group  
Department of Information Technologies and Systems  
Indra-UCLM Research and Development Institute  
University of Castilla-La Mancha  
Paseo de la Universidad, 4 – 13071 Ciudad Real, Spain  
{JoseAntonio.Cruz,Marcela.Genero,Mario.Piattini}@uclm.es

**Abstract.** In this work, we present the main conclusions obtained from the definition and validation of a set of measures for UML statechart diagrams, in a methodological way. The main focus is the empirical validation of the measures as early understandability indicators.

## 1 Introduction

The availability of valid measures in the early phases of the software development life-cycle allows a better management of the later phases. The measures allow the designers a quantitative comparison of design alternatives, and therefore an objective selection among several conceptual modelling alternatives with equivalent semantic content. Besides, designers can predict external quality characteristics, like maintainability, in the initial phases of the software life-cycle and better allocate the resources based on these predictions.

In order to define valid measures, we have followed and refined a process for measure definition [7] that consists of three main steps: measure definition, and theoretical and empirical validation. This process pays especial emphasis on some issues that must be taken into account when defining measures for software, such as:

- Measures must be well-defined, pursuing clear goals.
- Measures must be theoretically validated, by addressing the question ‘is the measure measuring the attribute it is purporting to measure?’
- Measures must be empirically validated, by addressing the question ‘is the measure useful in the sense that it is related to other external quality attributes in the expected way?’
- Measures must be defined in a precise way, avoiding misconceptions and misunderstandings.
- Measures calculation must be easy and it is better if their extraction is automated by a tool.

In this work, we present the main results obtained after defining a set of measures keeping these issues in mind, paying special attention to the empirical validation of the measures.

Section 2 presents the informal and formal definition of the measures. Section 3 tackles their theoretical validation. Section 4 explains the different families of experiments performed for achieving a thorough empirical validation of the measures. Section 5 provides the main features of a tool developed for the automatic calculation of the measures. Finally, section 6 summarizes the main conclusions achieved and outlines the future work.

## 2 Measures Definition

The main concern of this research was the definition of a set of early indicators of the understandability of UML statechart diagrams. But understandability, as an external quality attribute, is hard to measure early in the modelling process. Therefore, an indirect measurement based on internal properties of the model such as the structural complexity, was required [6].

The main concern of the measures definition step is explaining what the measures intend to measure. In the recent years, a great number of measures proposals for OO software products have been developed but most of them present a lack of formalization in their definition. This fact leads a set of difficulties to arise such as [1]:

- Experimental findings can be misunderstood due to the fact may be not clear what the measure really captures.
- Measures extraction tools can arrive to different results.
- Experiments replication is hampered.

Our work has refined the referred method by formally defining the measures using two formal languages: OCL [17] and Maude [9]. Table 1 presents the informal

**Table 1.** Definition of the measures in natural language

Measure	Accron.	Description
Nesting Level in Composite States	NLCS	The maximum number of composite states nested within other composite states in the statechart diagram.
Number of Activities	NA	The total number of activities in the statechart diagram.
Number of Composite States	NCS	The total number of composite states in the statechart diagram.
Number of Complex Transitions	NCT	A complex transition has attached a number of guard conditions, events or actions, while a simple transition does not.
Number of Events	NE	The total number of events, whichever the type they are.
Number of Entry Actions	NEntryA	The total number of entry actions, i.e., the actions performed each time a certain state is reached.
Number of Exit Actions	NExitA	The total number of exit actions, i.e., the actions performed each time a certain state is left.
Number of Guards	NG	The total number of guard conditions of the statechart diagram.
Number of Indirect Actions	NIA	Number of actions to be performed associated to transitions.
Number of Simple States	NSS	The total number of states, also considering the simple states within the composite states.
Number of Transitions	NT	The total number of transitions, considering common transitions, the initial and final transitions, self-transitions and internal transitions

```

context StateMachine::NCS():Integer
body: result = self.top.allSubstates()->collect(s |
                s.ocIsTypeof(CompositeState))-> size()

```

**Fig. 1.** Formal definition of the NCS measure using OCL

```

...

op NIA : StatechartDiagram -> Int .
op analyzeTA : TransitionList -> Int .

...

eq NIA(statechartDiagram (S, TL)) = analyzeTA(TL) .

...

eq analyzeTA(noTransition) = 0 .
eq analyzeTA(transition(TN, NEVNL, NEVNL2, noTransitionLabel)) = 0 .
eq analyzeTA(transition(TN, NEVNL, NEVNL2, transitionLabel(E, GC, AEL))) =
    if (AEL == noActionExpression) then
        0
    else
        length(AEL)
fi .

eq analyzeTA(transition(TN, NEVNL, NEVNL2, noTransitionLabel) TL ) =
    analyzeTA(TL)
eq analyzeTA(transition(TN, NEVNL, NEVNL2, transitionLabel(E, GC, AEL))T ) =
    if (AEL == noActionExpression) then
        analyzeTA(TL)
    else
        length(AEL) + analyzeTA(TL)
fi .

...

```

**Fig. 2.** Formal definition of the NIA measure using Maude

definition of the measures in natural language whilst Fig. 1 provides an example of the formal definition of the measure NCS using OCL and Fig. 2 provides an example of the formal definition of the measure NIA using Maude.

### 3 Theoretical Validation

The theoretical validation was carried out to show that a measure is really measuring the attribute that it aims to measure [2]. Moreover, it provides information related to the mathematical and statistical operations that can be done with the measures, e.g., the scale in which a measure should be measured.

In our work, we have based on the property-based framework proposed by Briand [3, 4] and the Measurement Theory-based DISTANCE framework [18]. The first one

has characterized the measures as size or complexity measures, while the second framework has characterized all the measures as ratio scale.

4 Empirical Validation

Empirical validation is an on-going activity [2] performed to demonstrate the usefulness of a measure. This phase is necessary before any attempt to use measures as objective and early indicators of quality.

In this section, we will describe the three families of experiments that have been carried out during our work. In section 4.1, we will explain a family of three experiments performed in order to empirically validate the measures proposed in this PhD thesis and build a preliminary understandability prediction model by means of a regression analysis using a technique specifically recommended when the data had been obtained through a repeated measures design.

The second empirical study, described in section 4.2, is another family composed by five different empirical studies. This family was used for assessing how composite states affected the understandability of UML statechart diagrams.

The third family, explained in section 4.3, is composed by a controlled experiment and a replication of it that were performed in order to study the optimal nesting level of composite states within the UML statechart diagrams.

Fig. 3 gives a global vision of the whole empirical validation process.

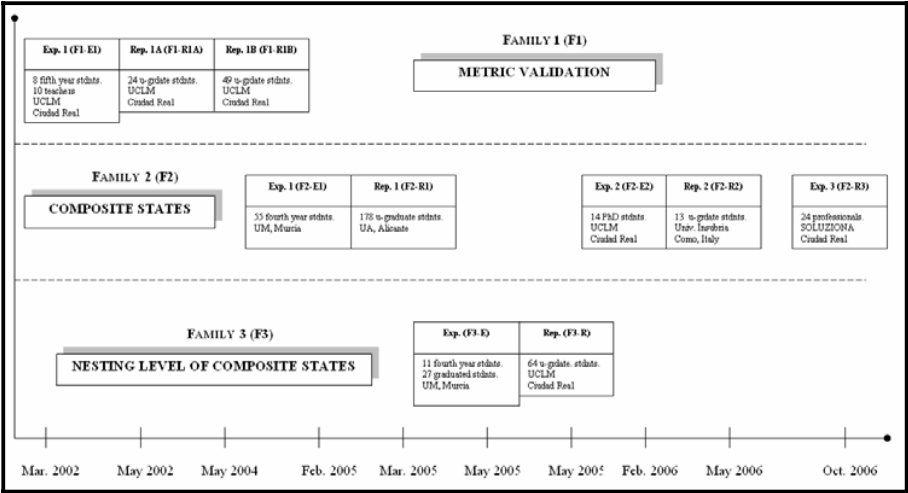


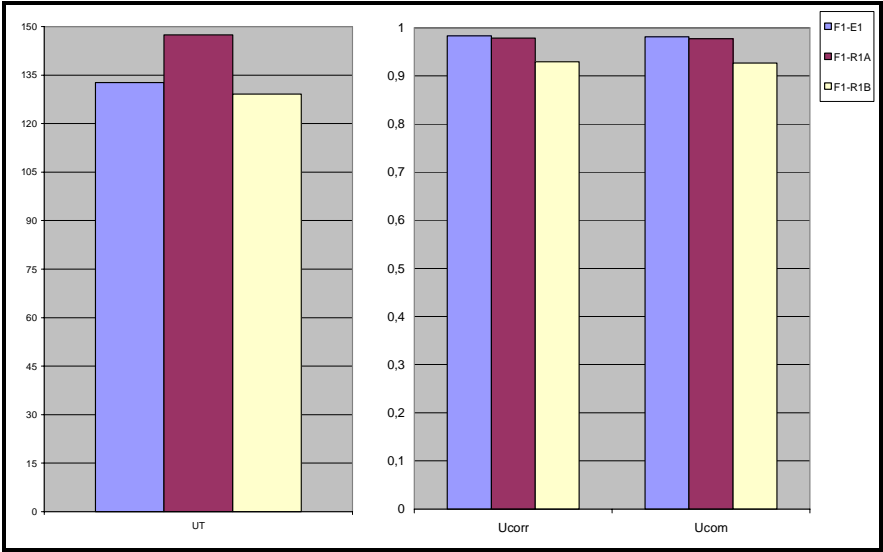
Fig. 3. Chronology of the families of experiments

4.1 First family of Experiments (F1): Performing the Measures Validation

This first family of experiments was performed for studying the relationship between the different defined measures and the understandability of UML statechart diagrams. The main characteristics of this family can be found in Table 2, while Fig.4

**Table 2.** Characteristics of the family F1

<b>Context Definition</b>	<b>Subjects</b>	E1: 8 PhD students, 10 teachers R1A: 24 students R1B: 49 students
	<b>Location</b>	University of Castilla-La Mancha (Spain)
	<b>Date</b>	E1: March 2002 R1A: May 2002 R1B: May 2004
<b>Materials</b>		20 diagrams with different values for the measures
<b>Variables Definition</b>	<b>Dependent</b>	Understandability of UML statechart diagrams, measured by UT (time), UCorr (correctness) and UCom (completeness), later by UEffc (efficiency).
	<b>Independent</b>	Measures for UML statechart diagrams understandability
<b>Hypotheses</b>		H <sub>0,1</sub> : There is not a significant correlation between the UML statechart diagrams measures and the understandability time. H <sub>0,2</sub> : There is not a significant correlation between the UML statechart diagrams size measures and understandability correctness. H <sub>0,3</sub> : There is not a significant correlation between the UML statechart diagrams size measures and understandability completeness.



**Fig. 4.** Average values for UT, UCorr and UCom in F1

graphically shows the average values that were obtained for the different measurements related to the dependent variable.

We could observe that while for UCorr and UCom, the values seemed quite logical and the teachers and higher-degree students had obtained better results, we did not obtain those results with the UT. This fact made us think that time, on its own, was not a good indicator of the understandability of the diagrams, so we used the understandability efficiency, which relates the number of correct answers with the time invested

answering the questions. The results with this new measure agreed with those obtained for UCorr and UCom.

Later, we performed a Spearman’s correlation analysis and obtained that the measures NA, NSS, NG and NT were highly correlated with the understandability efficiency of the diagrams.

We also performed a PCA that characterized the measures into three different groups:

- Simple States Features (SSF), composed by measures that explore the relationships between the different states of the diagrams and also the states themselves.
- Activities within States (AWS), composed by the measures of the activities performed after entering or leaving a state.
- Number of Activities (NA), composed only by this measure.

Finally we built the preliminary regression model for the understandability efficiency shown next:

$$UEff = 0.011575 - 0.000813*NA - 0.000204*SSF - 0.000273*AWS \tag{1}$$

Further details about this first family of experiments can be found in [13].

4.2 Second Family of Experiments (F2): Studying the Composite States

In the previous study, the composite states did not show a clear effect on the understandability of the diagrams, so we decided to study them specifically. The main characteristics to the different studies performed in this family are shown in Table 3.

The main strength of this family relies on the evolution of materials, tasks, and subjects that it has suffered along its performance.

First, we noticed that the materials used were not complicated enough to obtain actual results, so we increased their difficulty from the different experiments until using a real-project model in the last experiment (E3) [21].

Table 3. Characteristics of the family F2

Context Definition	Subjects	E1: 55 students R1:178 students E2: 14 PhD students R2: 13 students E3: 24 professionals
	Location	E1: University of Murcia R1: University of Alicante E2: University of Castilla-La Mancha R2: Università dell’Insubria E3: SOLUZIONE SW-Factory
	Date	E1: February 2005 R1: March 2005 E2:February 2006 R2: May 2006 E3: October 2006
Variables Definition	Dependent	Understandability of UML statechart diagrams, measured by UEffec (always) and UReten and UTrans (E2, R2 & E3).
	Independents	The use of composite states in the diagram (always) and the domain of the diagram (E1, R1, E2 & R2).
Hypotheses		H1a: using composite states improves UEffec in subjects when trying to understand an UML statechart diagram. (always) H1b: using composite states improves UTrans in subjects when trying to understand an UML statechart diagram. (E2, R2 & E3) H1c: using composite states improves UReten in subjects when trying to understand an UML statechart diagram. (E2, R2 & E3)

The original tasks also evolved by the use of the Cognitive Theory of Multimedia Learning [16], which provided the measures of transfer and retention for measuring the dependent variable.

Finally, we used students in the first studies but in the last one, we counted on a set of real practitioners in order to alleviate the possible lack of experience that the students might have.

The conclusions of this family indicate that the use of composite states does not significantly improve the understandability of UML statechart diagrams, at least when working with diagrams whose size and complexity are not too high.

Further details about this second family of experiments can be found in [11, 12].

### 4.3 Third Family of Experiments (F3): Looking for the Nesting Level of Composite States

The use of hierarchical structures and how they affected the quality of different modeling techniques has been broadly studied [5, 8, 14, 19, 20]. In the same direction that most of these works, we intended to assess which was the optimal level of inheritance within a composite state in an UML statechart diagram. This was the aim of this third family of experiments that we performed. Its main characteristics are detailed in Table 4.

**Table 4.** Characteristics of the family F3

<b>Context Definition</b>	<b>Subjects</b>	Exp: 38 students Rep: 64 students
	<b>Location</b>	Exp: University of Murcia Rep: University of Castilla-La Mancha
	<b>Date</b>	May 2005
<b>Materials</b>		3 diagrams with values 0, 1 and 2 for the measure NLCS
<b>Variables Definition</b>	<b>Dependent</b>	Understandability of UML SD, measured by UCorr and UEffic.
	<b>Independent</b>	Nesting Level within composite states in an UML SD
<b>Hypotheses</b>		$H_{0-ij}$ : the understandability of UML statechart diagrams with $i$ and $j$ composite states nesting levels is not significantly different $H_{1-ij}$ : the understandability of UML statechart diagrams with $i$ and $j$ composite states nesting levels is significantly different In both cases, $i, j \in \{0, 1, 2\}$ and $i \neq j$ .

The results obtained indicating that a flat nesting level within composite states made the diagrams more understandable.

More details about this third family of experiments can be found in [10].

## 5 GenMETRIC

GenMETRIC [15] is a tool for defining, calculating and visualizing software measures. This tool supports the management of the measurement process by supporting the definition of measurement models, the calculation of the measures defined in

those measurement models and the presentation of the results in tables and graphically.

The two key characteristics of GenMETRIC are:

- **Genericity.** With this tool it is possible to measure any software entity. The requirement necessary to achieve this goal is that the metamodel representing the software entity (domain metamodel) must be included in the repository of the tool. The different measures must be defined on the elements of the domain metamodels. This implies that in order to measure new entities it is not necessary to add a new code to GenMETRIC.
- **Extensibility.** GenMETRIC supports the definition of any software measure. The base measures are defined on the domain metamodel elements (classes and associations) by using standard measurement methods such as “count” or “graph length”. For the definition of derived measures and indicators the tool includes an evaluator of arithmetical and logical expressions, as Fig. 5 shows.

**New Measure Definition**

**Name** Number of Guards **Acronym** NG

**Description** Calculates the total number of guard conditions in the statechart diagram

**Metamodel** UML Statechart Diagrams

☒ Base Measure ☐ Derived Measure

Measurement Method	Metamodel Elements
Count	Composite State
Maximum graph length	Simple State
	Transition
	Event
	Guard
	Actions
	Entry Activities
	Exit Activities

**Derived Measure Definition**

Defined Measures

Measurement Function

+ - \* /

Add Clear Close

**Fig. 5.** New measure definition in GenMETRIC

## 6 Conclusions

In this work, we have illustrated the relevance of defining valid measures in a methodological way, following three main steps:



- **Measure Definition.** We defined the measures with the idea of finding indicators of the understandability of UML statechart diagrams. Firstly, we defined them in natural language. Later, we have formally defined the measures using OCL and Maude, in order to alleviate a set of problems well-known by the Software Engineering community.
- **Theoretical Validation.** We have also theoretically validated the measures using two different approaches, one based on properties and another based on the Measurement Theory.
- **Empirical Validation.** We have presented three different families of experiments that were carried out in order to empirically check the validity of the measures previously presented. The first family provided as a result that a group of the measures were highly correlated with the understandability efficiency of UML statechart diagrams, as well as a preliminary prediction model for the understandability efficiency. The second family studied the effect that composite states have on the understandability of UML statechart diagrams. The results obtained indicate that, quite surprisingly, these structures do not improve significantly the understandability of the diagrams, at least in the conditions that we have used in our experimentation process. Agreeing with the previous family, the third family has allowed us conclude that using a flat nesting level within composite states makes an UML statechart diagram more understandable.

Finally, we have introduced GenMETRIC, a generic and extensible tool for the definition and automatic calculation of the different measures.

As future work, we are aware that it is necessary to continue refining the proposed measures and even to define some new ones (if necessary). This way we could have an adequate and useful set of measures for measuring the quality characteristics of UML statechart diagrams. Further empirical studies must also be preformed in order to reach a definitive and strong empirical validation of all the measures.

We will also try to extend the definition of measures to some other quality characteristics, such as modifiability, that also affect the maintainability of the diagrams.

## Acknowledgements

This research is part of the MECENAS and the IDONEO projects (PBI06-0024, PAC08-0160-6141) financed by “Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha” and the ESFINGE project (TIN2006-15175-C05-05) (TIN20005-24055-E) supported by the “Ministerio de Educación y Ciencia” (Spain).

## References

1. Baroni, A.L., Braz, S., Brito e Abreu, F.: Using OCL to Formalize Object-Oriented Design Metrics Definitions. In: Proceedings of 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002), Malaga, Spain, pp. 99–106 (2002)
2. Briand, L., El-Emam, K., Morasca, S.: Theoretical and Empirical Validation of Software Product Measures, ISERN (1995)

3. Briand, L., Morasca, S., Basili, V.: Property-Based Software Engineering Measurement. *IEEE Transactions on Software Engineering* 22(1), 68–86 (1996)
4. Briand, L., Morasca, S., Basili, V.: Response to: Comments on Property-Based Software Engineering Measurement: Refining the Additivity Properties. *IEEE Transactions on Software Engineering* 23, 196–197 (1997)
5. Briand, L., Wüst, J., Daly, J., Porter, V.: Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems. *The Journal of Systems and Software* 51, 245–273 (2000)
6. Briand, L., Wüst, J., Lounis, H.: Investigating Quality Factors in Object-oriented Designs: An Industrial Case Study, Technical Report ISERN 98-29, version 2 (1998)
7. Calero, C., Piattini, M., Genero, M.: Method for Obtaining Correct Metrics. In: *Proceedings of 3rd International Conference on Enterprise and Information Systems (ICEIS 2001)*, Setúbal, Portugal, pp. 779–784 (2001)
8. Cartwright, M.: An Empirical View of Inheritance. *Information and Software Technology* 40(4), 795–799 (1998)
9. Clavel, M., Durán, F., Eker, S., Lincoln, P., Marínez-Oliet, N., Meseguer, J., Talcott, C.: *Maude 2.1.1 Manual*, University of Illinois at Urbana-Champaign (2003)
10. Cruz-Lemus, J.A., Genero, M., Piattini, M.: Investigating the Nesting Level of Composite States in UML Statechart Diagrams. In: *Proceedings of 9th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2005)*, Glasgow, United Kingdom, pp. 97–108 (2005)
11. Cruz-Lemus, J.A., Genero, M., Piattini, M., Morasca, S.: Improving the Experimentation for Evaluating the Effect of Composite States on the Understandability of UML Statechart Diagrams. In: *Proceedings of 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2006)*, Rio de Janeiro, Brazil, pp. 9–11 (2006)
12. Cruz-Lemus, J.A., Genero, M., Piattini, M., Toval Álvarez, J.A.: An Empirical Study of the Nesting Level of Composite States within UML Statechart Diagrams. In: Akoka, J., Liddle, S.W., Song, I.-Y., Bertolotto, M., Comyn-Wattiau, I., van den Heuvel, W.-J., Kolp, M., Trujillo, J., Kop, C., Mayr, H.C. (eds.) *ER Workshops 2005*. LNCS, vol. 3770, pp. 12–22. Springer, Heidelberg (2005)
13. Cruz-Lemus, J.A., Maes, A., Genero, M., Poels, G., Piattini, M.: The Impact of Structural Complexity on the Understandability of UML Statechart Diagrams, University of Ghent (2007)
14. Daly, J., Brooks, A., Miller, J., Roper, M., Wood, M.: An Empirical Study Evaluating Depth of Inheritance on Maintainability of Object-Oriented Software. *Empirical Software Engineering* 1(2), 109–132 (1996)
15. García, F., Serrano, M., Cruz-Lemus, J.A., Ruiz, F., Piattini, M.: Managing Software Process Measurement: a Metamodel-based Approach. *Information Sciences* 177, 2570–2586 (2007)
16. Mayer, R.E.: *Multimedia Learning*, Cambridge University Press (2001)
17. OMG, UML 2.0 OCL Final Adopted Specification, Object Management Group (2005)
18. Poels, G., Dedene, G.: Distance: A Framework for Software Measure Construction. Department of Applied Economics, Catholic University of Leuven, Belgium (1999)
19. Poels, G., Dedene, G.: Evaluating the Effect of Inheritance on the Modifiability of Object-Oriented Business Domain Models. In: *Proceedings of 5th European Conference on Software Maintenance and Reengineering (CSMR 2001)*, Lisbon, Portugal, pp. 20–29 (2001)
20. Prechelt, L., Unger, B., Philippsen, M., Tichy, W.: A Controlled Experiment on Inheritance Depth as a Cost Factor for Code Maintenance. *The Journal of Systems and Software* 65, 115–126 (2003)
21. Webb, K.: *Xholon Digital Watch Project* (2006)

# Adapting the COSMIC Method for Evaluating the Functional Size in PRiM

Gemma Grau

Universitat Politècnica de Catalunya (UPC)  
c/ Jordi Girona 1-3, Barcelona E-08034, Spain  
ggrau@lsi.upc.edu

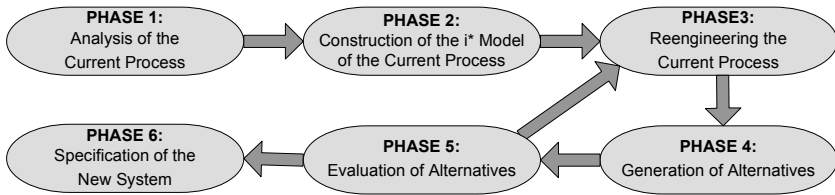
**Abstract.** The COSMIC method is a standard that has been proven effective for measuring the functional size of business applications and real-time software systems from their functional user requirements specification. Despite of this, the approaches based on the COSMIC method usually require a mapping between the concepts in the requirements specification and their own terms and do not take into account non-functional requirements. On the other hand, PRiM is a method that aims at assessing non-functional properties at the early stages of the development process. PRiM uses the  $i^*$  framework to model the functional and non-functional requirements in terms of actors and dependencies among them. In this paper we present how the  $i^*$  constructs proposed in PRiM can be adapted to measure the functional size using the COSMIC method and, as PRiM works with requirements and allows the evaluation of non-functional properties, there is a remarkable benefit when using both methods altogether.

## 1 Introduction

The COSMIC method [1] provides a set of principles for measuring the functional size, and it is supported by the ISO/IEC 19761:2003 [16]. The functional size is measured based on the functional user requirements specification of the software systems and can be applied over the domains of business applications and real-time software systems. Its effectiveness has converted it into a well-establish method and many measurement procedures have arisen to support it (for instance, [2], [4], [13], [19]). However, there still some open points. First, as the COSMIC method aims at measuring the functional size, non-functional properties are not taken into account. Second, there is a lack of a clear definition of the various concepts that contribute to software size, particularly at the requirements modelling level [5]. Finally, it requires the construction of a particular artefact and, as stated in [13], for productivity reasons it is important to avoid model reconstruction dedicated for just measuring purposes.

On the other hand, PRiM [9] is a Process Reengineering  $i^*$  Method that addresses the specification, analysis and design of information systems from a reengineering point of view. PRiM is based on the premise that, nowadays, most of the information systems are not built from the scratch, but from a legacy system or a human process that need to be reengineered. PRiM uses the  $i^*$  framework [23] for representing the software model of the system in terms of actors and dependencies between them. The PRiM method is conformed by the six phases presented in Fig. 1. The first phase

involves capturing and recording the information about the current process in order to inform further phases. During the second phase, the  $i^*$  model of the current process is built. In order to reengineer the current process, new goals are obtained, which is done in the third phase of the method. With the aim of satisfying these goals, several process alternatives are systematically generated during the fourth phase. In the fifth phase, the different alternative  $i^*$  models are evaluated by applying structural metrics over them [8]. Finally, in the sixth phase, PRiM proposes the generation of the new information system specification from the  $i^*$  model of the chosen alternative. We remark that these phases can be iterated and intertwined as needed as long as the outputs of one phases can be used as inputs for the next one.



**Fig. 1.** Overview of the PRiM method

The structural metrics proposed by PRiM focus on the evaluation of non-functional properties such as ease of use, process agility, maintainability, etc. However, despite that the  $i^*$  models are constructed taking into account the functional requirements of the system, functional metrics are not considered by the method. In [10] we propose a framework to include different techniques within a reengineering framework such as PRiM. In that context we consider adequate to adapt the PRiM method to measure the functional size because we have observed strong similarities between the representation of the concepts used in the COSMIC method and the ones represented in PRiM. Thus, we propose to calculate functional size to complement the set of metrics proposed in PRiM, and to use the tool support provided by J-PRiM [12]. As the PRiM method provides techniques for the elicitation of requirements, the generation of design alternatives and the evaluation of non-functional properties, we believe that the use of both techniques altogether provides mutual benefits.

In order to verify that the COSMIC method can be correctly applied within the PRiM context, we have applied the measurement process steps proposed in [17], which are summarized into four main steps: 1) Design of the process method, 2) Application of the measurement method rules, 3) Analysis of the measurement result, and 4) Exploitation of the measurement result. However, due to the lack of space, here we only present the firsts two steps. More precisely, for the design of the process method we focus on the mapping between the concepts of PRiM and the ones of the COSMIC method metamodel, and on the definition of the numerical assignment rules. On the other hand, for the application of the measurement method we focus on how the software documentation is obtained and the software model is constructed using the J-PRiM tool, and on how we apply the numerical assignment rules using the structural metrics. The proposed process is exemplified by using the C-Registration Case Study [20].

The remainder of this paper is organized as follows. In section 2 we introduce the  $i^*$  framework. As a first step for using the COSMIC method within PRiM, in section 3, we present the mapping between the modelling and evaluation concepts in both methods. In section 4 we show how the functional size is measured in PRiM and, in section 5, we introduce how non-functional properties can be evaluated within this method. In section 6 we compare the presented approach with related work and, finally, in section 7 we present the conclusions and future work.

## 2 The $i^*$ Framework

The  $i^*$  framework is a goal-oriented language defined by Eric Yu [23] with the aim of modelling and reasoning about organizational environments and their information systems. For doing so, it offers a formal representation of the involved actors and their behaviours allowing the consideration of both functional and non-functional requirements. The  $i^*$  framework proposes the use of two types of models for modelling systems, each one corresponding to a different abstraction level: a Strategic Dependency (SD) model represents the strategic level by means of the dependencies between the actors, whilst the Strategic Rationale (SR) model represents the rational level by means of showing the intentionality inside each one of the represented actors. As the COSMIC method takes into account the interaction between the actors rather than in its internal behaviour, in this work we focus on the SD model.

A SD model consists of a set of nodes that represent actors and a set of dependencies that represent the relationships among them, expressing that an actor (*depender*) depends on some other (*dependee*) in order to obtain some objective (*dependum*). The *dependum* is an intentional element that can belong to one of the following four types: goal, task, resource, and softgoal. The semantics are:

- For goal dependencies, the *dependee* is free to make whatever decisions are necessary to achieve the goal. For instance, in Fig. 2, the *Registrar* depends on the *C-Registration System* for the goal *Student information is maintained*.
- For task dependencies, the *depender* depends upon the *dependee* to attain a goal following a prescriptive procedure. For instance, in Fig. 2, the *Professor* depends on the *C-Registration System* to *Submit student grades*, which has to be done following its own procedure.
- For resource dependencies, the *depender* depends upon a *dependee* for the availability of a physical or informational entity. For instance, in Fig. 2, the *C-Registration System* depends on the *Student* to obtain the entity *Course registration*.
- For softgoal dependencies, the *depender* depends upon the *dependee* to attain some goal, perform some task, or produce some resource, in a particular way. The *Student* depends on the *C-Registration System* for a *Secure remote access to the system*.

The graphical notation is shown in Fig. 2. For more details we refer to [23].

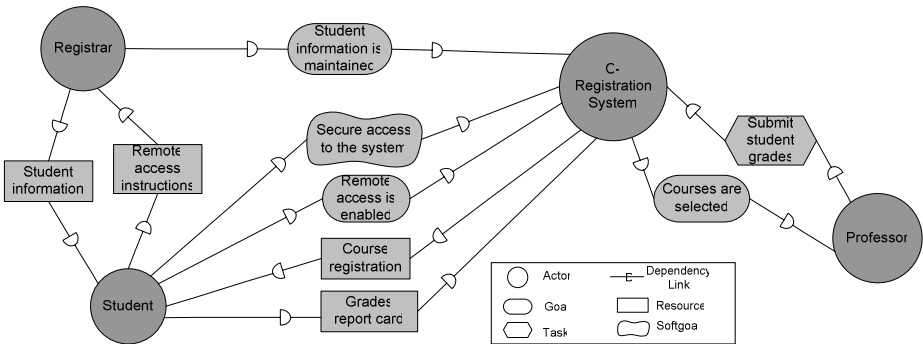


Fig. 2. Excerpt of an  $i^*$  model for the C-Registration System

### 3 Mapping Phase: From COSMIC to PRiM

The first of the measurement process steps proposed in [17] refers to the design of the process method, which includes the following substeps: 1) Definition of the objectives; 2) Characterization of the concept to be measured; 3) Design or selection of a metamodel for the object to be measured; and, 4) the definition of the numerical assignment rules. We remark that the first two steps are already addressed in the premises of this work. More precisely, for the definition of the objectives, we establish that the concept to be measured is the functional size and, the characterization of the concept to be measured is the one stated in the COSMIC method. Therefore, in this section we address the selection of the metamodel and the definition of the numerical assignment rules, focusing on the mapping between the concepts proposed by the COSMIC method and PRiM.

#### 3.1 Design or Selection of the Metamodel

According to [17] the set of characteristics selected to represent the software and the set of their relationships, constitute the metamodel proposed for the description of the software to which the proposed measurement method will be applied. As we want to apply the COSMIC functional size to  $i^*$  models generated with PRiM, the metamodel selected for representing the software model to be evaluated is the PRiM  $i^*$  metamodel, which allows a mapping between the COSMIC and the  $i^*$  concepts. In order to understand this mapping we first present an overview of both methods.

**The PRiM Method.** PRiM is composed by six different phases (see Fig. 1), starting from the analysis of the current situation and its representation using the  $i^*$  framework. In PRiM, the  $i^*$  model is constructed in two different processes in order to differentiate the functionality that is performed by the system (Operational  $i^*$  Model) from the strategic needs of the organization (Intentional  $i^*$  Model). The Intentional  $i^*$  Model takes into account non-functional requirements and, as we are interested in the functional user requirements, here we only address the Operational  $i^*$  Model. The Operational  $i^*$  Model is constructed based upon the information available from the current process or on the description of how the new process has to be. In order to

facilitate the further construction of the Operational  $i^*$  Model, this information is summarized into Detailed Interaction Scripts (DIS). DIS are scenario-based templates that describe the information of each activity of the current process by means of its preconditions, postconditions, triggering events, and a list of the actions undertaken in the activity. For each action, it specifies the actor that initiates the action (*initiator*), the name of the action, the resource involved (differentiating if is produced, provided, or consumed by the *initiator*) and the actor to which the action is addressed (*addressee*). PRiM does not enforce any scenario-based technique for filling the DIS templates and so, it is possible to apply use cases or any other scenario-based technique for documenting the current process as long as it follows the structure proposed. Once the process is described using the DIS templates, the PRiM method provides precise rules that allows to transform the information on the DIS to the Operational  $i^*$  Model, which can be done automatically with the tool support provided by J-PRiM [12].

**The COSMIC Method.** The COSMIC measurement method defines a measure of software functional size that is standardized in the ISO /IEC 19761:2003 [16]. The COSMIC method involves applying a set of models, principles, rules and processes to the Functional User Requirements (FUR) of a given piece of software, by following three phases: the measurement strategy, the mapping phase, and the measurement phases. The result is a size measure expressed in COSMIC Function Points (CFP). Functional User Requirements are a subset of the user requirements, which describe what the piece of software to be measured shall do in terms of tasks and services. Functional User Requirements are defined based on the principles that there is a set of so-called Functional Users of the software to be measured that are senders and/or intended recipients of data. The software interacts with these Functional Users via data movements across a conceptual interface called boundary, and it also moves data to and from Persistent Storage, also across the boundary.

In the COSMIC Generic Software Model, a Functional Users Requirements are mapped into unique functional process, where each functional process consists of subprocesses, which can be a data movement or a data manipulation. A data movement moves a single data group, and there are four types of data movement: an Entry moves a data group into the software from a functional user; an Exit moves a data group out of the software to a functional user; a Write moves a data group from the software to persistent storage; and, a Read moves a data group from persistent storage to software. Fig. 3 shows the components of a functional process and the explained relationships. COSMIC defines that, for any functional process, the functional sizes of individual data movements shall be aggregated into a single functional size value in units of CFP by arithmetically adding them together.

$$\text{Size (functional process)} = \sum \text{size}(\text{Entries}_i) + \sum \text{size}(\text{Exits}_i) + \sum \text{size}(\text{Reads}_i) + \sum \text{size}(\text{Writes}_i)$$

Table 1 presents the definitions for the COSMIC concepts and establishes an analogy with the  $i^*$  framework. We remark that in order to establish this analogy, we assume that it is possible to classify the  $i^*$  actors as: Functional User (FU), Functional Process (FP), and Persistent Storage (PS). The COSMIC functional size is calculated by assigning to each data movement, a single unit of measure which is, by convention, equal to 1 CFP (COSMIC Function Point). Therefore, the total size of the

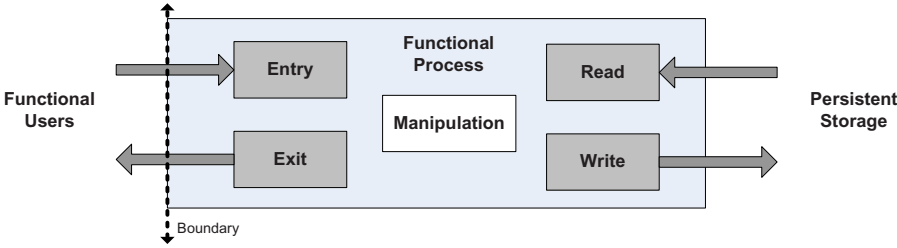


Fig. 3. The components of a functional process and some of their relationships, [1]

Table 1. Mapping of the COSMIC method concepts to the *i\** concepts

COSMIC concept (obtained from [1])		<i>i*</i> concept
Functional User	A (type of) user that is a sender and/or and intended recipient of data in the Functional User Requirements of a piece of software.	Actor that represents one or more human roles that have functional dependencies over the software under study.
Functional Process	Elementary component of a set of Functional User Requirements comprising a unique, cohesive, and independently executable set of data movements.	Actor that models the component that performs the functionality that is considered for the measurement.
Persistent Storage	Storage which enables a functional process to store or retrieve a data group beyond the life of the functional process.	Actor that represents the entities that manage data in a persistent way.
Data Group	Is a distinct, no empty, non ordered and non-redundant set of data attributes where each included data attribute describes a complementary aspect of the same object of interest.	Resource element that represents a physical or informational entity.
Data Movement	A base functional component which moves a single data group type.	Any dependency where the <i>dependum</i> is a resource.
Entry (E)	Data movement type that moves a data group from a functional user across the boundary into the functional process where it is required.	<i>Dependum</i> : Resource (data group) <i>Depender</i> : Functional Process <i>Dependee</i> : Functional User
Exit (X)	Data movement type that moves a data group from a functional process across the boundary to the functional user that requires it.	<i>Dependum</i> : Resource (data group) <i>Depender</i> : Functional User <i>Dependee</i> : Functional Process
Read (R)	Data movement type that moves a data group from persistent storage within reach of the functional process which requires it.	<i>Dependum</i> : Resource (data group) <i>Depender</i> : Functional Process <i>Dependee</i> : Persistent Storage
Write (W)	Data movement type that moves a data group lying inside a functional process to persistent storage.	<i>Dependum</i> : Resource (data group) <i>Depender</i> : Persistent Storage <i>Dependee</i> : Functional Process

software being measured corresponds to the addition of all data movements recognized by the COSMIC method.

From the information on Table 1, we observe some similarities between the COSMIC measurement process model and PRiM. Thus, in Fig. 4, we have established a set of mapping relationships between the concepts needed in the Functional User Requirements (FUR) of the COSMIC generic software model, the information documented in the DIS tables of PRiM and the *i\** concepts. At the left of Fig. 4, we observe that the COSMIC method is based upon a *Functional Process* which has a



*Triggering Event* and several *Subprocesses* associated to it. Each *Subprocess* has a *Data Group* that can be of the type: entry (E), exit (X), read (R) or write (W). In the DIS, each *Functional Process* is represented by an *Activity*; the *Triggering Event* is part of the *Conditions* associated to the *Activity*; and, each *Subprocess* is represented by the concept of an *Action*. There is a correspondence between the concepts of *Data Group* and *Resource*, although the distinction between the *Data Group* types is implicit in the DIS information because it depends on the *Actors* that participate in the action. As we have already mentioned, PRiM proposes a set of automatic rules to transform DIS into *i\** Models (see [9] for details), where *Conditions* are transformed into *Goal Dependencies*; *Activities* and *Actions* are represented into SR elements; and, *Resource Dependencies* are established between the different *Actors*. In order to help the evaluation of the *i\** Model with the COSMIC method, we propose a classification of the *i\** actors into the following types: Functional User (FU), Functional Process (FP), and persistent storage (PS).

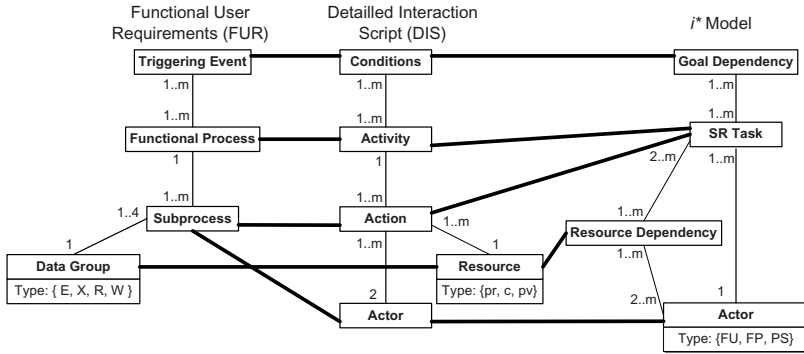


Fig. 4. Mapping across the different metamodels

### 3.2 Definition of the Numerical Assignment Rules

The PRiM method proposes structural metrics for evaluating the *i\** models and, in order to apply the COSMIC method, the numerical assignment rules are defined using the formulas and concepts proposed in [8], [9] which differentiate between actor-based and dependency-based functions. As in the COSMIC method the unit of measurement are the Data Groups, and in *i\** Data Groups are represented as dependencies. We are interested in dependency-based functions, which can be defined as follows.

**Dependency-Based metric.** Given a property  $P$  and an *i\** SD model that represents a system model  $M = (A, D)$ , where  $A$  is the set of the actors and  $D$  the dependencies among them, a dependency-based architectural metric for  $P$  over  $M$  is of the form:

$$P(M) = \frac{\sum_{d(a,b) \in D: \text{filter}_M(d) \times \text{correctionFactor}_M(a,b)} \text{limit}_P(D)}{\text{limit}_P(D)}$$

Where  $\text{filter}_M: D \rightarrow [0,1]$  is a function that assigns a weight to the every *dependum* (e.g., if the *dependum* is goal, resource, task, softgoal if it is from a specific kind), and

$\text{correctionFactor}_M: A \rightarrow [0,1]$  is a function that correct the weight accordingly to the kind of actor that the *dependor* and the *dependee* are, respectively. Finally,  $\text{limit}_P(D): A \rightarrow [1, \|A\|]$  is a function that normalizes the result obtained.

In order to measure the functional size, we have adjusted these factors according to the criteria established in Table 1. As we are interested in counting the total amount of CFP, we do not apply the normalization value and  $\text{limit}_P(D) = 1$ . Therefore, we define the metric as:

$$\text{Functional Size (M)} = \frac{\sum d: d \in D: \text{functional\_size}(d)}{\text{limit}_P(D)}$$

Where,

$$\text{filter}_M(d) = \begin{cases} 1, & \text{if } d \in \text{Resource} \\ 0, & \text{otherwise} \end{cases}$$

$$\text{correctionFactor}_M(a,b) = \begin{cases} 1, & \text{if } a \in \text{Functional Process and } b \in \text{Functional User (E)} \\ 1, & \text{if } a \in \text{Functional User and } b \in \text{Functional Process (X)} \\ 1, & \text{if } a \in \text{Functional Process and } b \in \text{Persistent Storage (R)} \\ 1, & \text{if } a \in \text{Persistent Storage and } b \in \text{Functional Process (W)} \\ 0, & \text{otherwise} \end{cases}$$

$$\text{limit}_P(D) = 1$$

## 4 Measurement Phase: Evaluating COSMIC with PRiM

Once the measurement method has been designed, the measurement process presented in [17] proposes three steps for its application: 1) software documentation gathering and construction of the software model; 2) application of the numerical assignment rules; and 3) measurement result analysis and exploitation of the result. As the final purpose of the measurement phase is to ensure that the defined mapping rules and the numerical assignment rules allow a reliable calculation of the functional size, we have apply the method to the three case studies presented at [7], and compliant to ISO 19761 [16]. For doing it, we have introduced the case studies in our tool J-PRiM [10], and we have obtained positive results with all of them. To illustrate the process, we present the C-Registration Case Study [20].

### 4.1 Software Documentation Gathering

The documentation of the software to be measured has to be collected before the application of the measurement method. For gathering the needed information, PRiM proposes to use Human Activity Models [18] to analyse the current process. On the other hand, COSMIC, proposes to derive Functional User Requirements from the software engineering artefacts that are produced before the software exists (for instance, requirements definition documents, or the results of the data or functional analysis of the requirements). Therefore, in both methods it is possible to measure the functional size prior to its implementation.

As we propose to use PRiM in our process, the documentation of the software interaction is done by completing the DIS templates. However, as we have proven that there is a mapping relationship between the COSMIC metamodel and the DIS metamodel, it is possible to use the COSMIC principles and rules to obtain the Functional User Requirements and, then, to document them using the DIS templates. In order to

illustrate the approach, we have chosen the C-Registration Case Study [20] and so, for the software documentation gathering, we use the Functional User Requirements provided in the case study in order to ensure that we are working with the same functional processes and data groups of the case study.

## 4.2 Construction of the Software Model

The software model describes how the software to be measured is represented by the measurement method. Therefore, the basis for its construction is the proposed meta-model. As we want to use the PRiM  $i^*$  metamodel, we document the Functional User Requirements using the DIS templates in order to use the rules that allow to automatically transform the information of the DIS templates into an Operational  $i^*$  Model.

Therefore, based on the Functional Processes provided in [20] we have established the activities of PRiM and, for each activity, we have described its actions by filling the DIS template. We have identified four different  $i^*$  actors: one software actor, which represents the *C-Registration System*; and three human actors which represent the *Registrar*, the *Professor*, and the *Student*. We remark that, in order to be compliant with the method, when describing the actions we have made explicit those actions that involve storing or retrieving information from the Persistent Storage. Although this was not considered in PRiM, it has been possible to introduce this information correctly from the problem statement provided. Fig. 5 shows the screenshot of J-PRiM when defining the activities for the C-Registration System Case

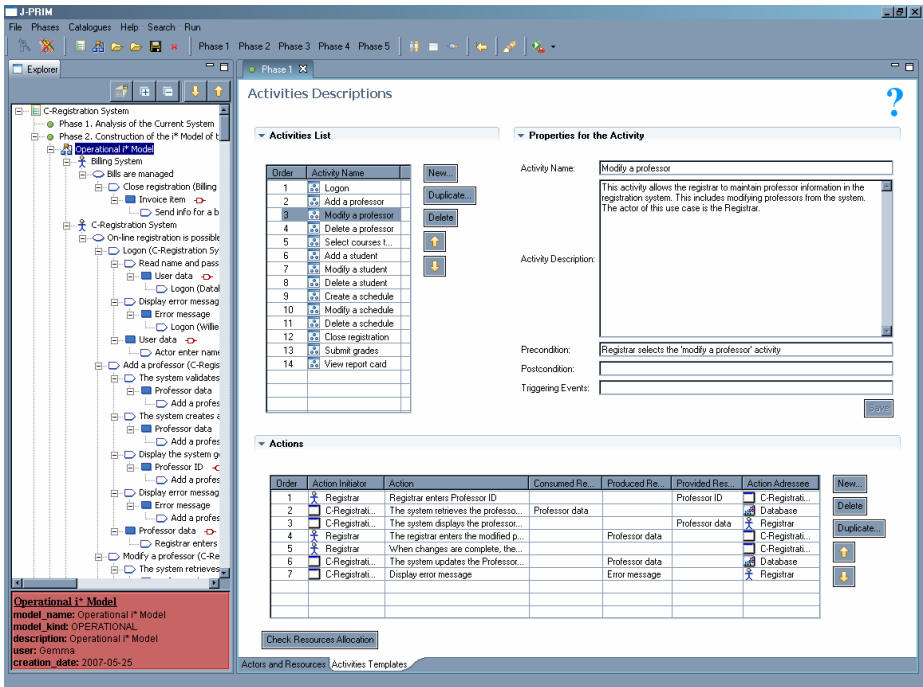


Fig. 5. Screenshot of J-PRiM: Activity definition and action description using the DIS template

Study. At the top we have the defined activities (at the left, the list of the activities and, at the right, its individual information). At the bottom we have the DIS template showing, for each action, the actor *initiator*, the resource involved (which can be consumed, produced or provided), and the actor *addressee*. At the left of Fig. 5 it is possible to see the representation of the Operational *i\** Model for the C-Registration System Case Study, which is generated automatically from the introduced information.

4.3 Application of the Numerical Assignment Rules

Once the Operational *i\** Model has been generated, we have applied the COSMIC dependency-based metric as defined in section 3.2. Fig. 6 presents the screenshot of the graphical interface that we have added in PRiM to facilitate the application of the COSMIC method. At the top-left side we show the different alternatives that can be evaluated, as the Operational *i\** Model contains all the information for calculating the functional size it is the one that has been selected. However, we remark that other Alternative *i\** Models could be generated according to the guidelines proposed in the PRiM method [9]. At the top-right side we have three boxes for classifying the actors according to the boundaries of the system (Functional User, Functional Process, and Persistent Storage). Finally, at the bottom, we can see the evaluation of each activity (or functional process), whilst the overall result is presented at the end of the list.

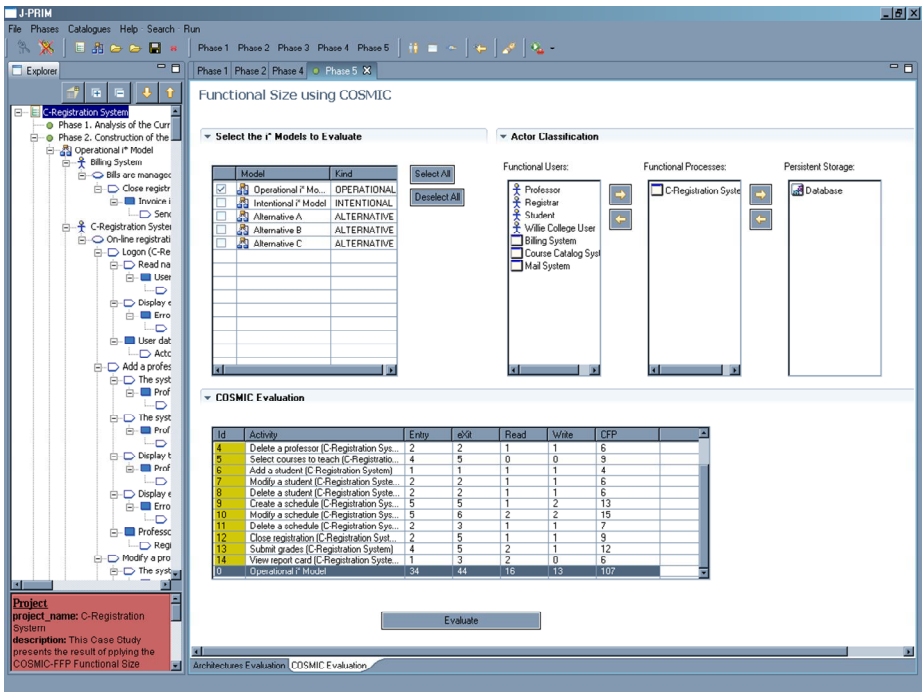


Fig. 6. Screenshot of J-PRiM: Evaluation of COSMIC for the selected Operational *i\** Model

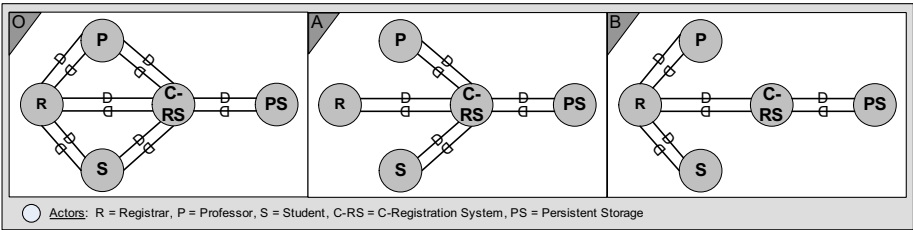
Once the result has been calculated, we have checked it with the result obtained in the case study. We have to mention that the first results were different from expected. In the first execution the reason was that, as we wanted to avoid copying the functional process data movements, we have generated our own action description. In this description we considered the command introduced by the Functional User as a triggering event, whilst in some functional processes of the case study this is considered as a data movement. We have added these data movements and we have regenerated the Operational  $i^*$  Model. In the second execution, the final result was 107 CFP, one unit higher than expected, which is due to a miscalculation on the final results presented in the C-Registration case study which scores 106 CFP.

## 5 Non-functional Measurements with PRiM

As we have previously mentioned, PRiM is a reengineering method that supports the generation and evaluation of alternatives. In PRiM the generation of alternatives is done based on the  $i^*$  model of the current process (namely the Operational  $i^*$  Model) and by reallocating the responsibilities between its actors. For instance, in the C-Registration Case Study, we can consider three different alternatives that are represented in Fig. 7 by means of its structural representation. We remark that the actors whose dependencies do not change between the alternatives are omitted (i.e. the Billing System, the Course Catalogue System and the Mail System).

- **Alternative O.** This alternative represents the current situation as described in the C-Registration Case Study [20]. In this alternative there are fourteen different functional processes (see list of activities in Fig. 5). In all the functional processes there is an interaction of a human actor with the C-Registration System software actor, being the human actor the responsible for introducing the needed information. For instance, the responsibility of ‘Modifying a student’ and ‘Modify a Professor’ falls on the *Registar* actor. Therefore, in Fig. 7, we observe that there are dependency relationships between the *Registar* and the *Student* and *Teacher*, as they provide their information to the *Registar* for its modification.
- **Alternative A.** This alternative considers the same allocation of responsibilities that in Alternative O, but we reallocate the responsibility of ‘Modify a student’ onto the *Student* actor, and the responsibility of ‘Modify a professor’ data onto the *Professor* actor. Thus, as presented in Fig. 7, we remark that there are no dependencies between the *Professor* and the *Student* with the *Registar*.
- **Alternative B.** This alternative considers that all the responsibility for interacting with the software, falls onto the *Registar* and, so, we consider that the student and the professor always request him for introducing their data. Therefore, in Fig. 7, there is no interaction between the *Student* and *Professor* actors and the *C-Registration System* actor, as the *Registar* is the one that access the software.

The PRiM method supports the generation of alternatives and their evaluation using structural metrics. The metrics proposed in PRiM can be organizational (dealing with non-functional properties of the organization) or architectural (dealing with non-functional properties of the software). For instance, in Table 2 the Operational  $i^*$



**Fig. 7.** Structural representations of the alternative  $i^*$  models of the C-Registration Case Study

Model and the two alternatives proposed are evaluated regarding to the organizational properties: *Ease of communication* and *Process agility*, as they are defined in [9]. When defining *Ease of communication*, we consider that the communication is easier when the two interacting actors are human, and it is damaged when it involves the software system. Under these premises, the results obtained for the C-Registration System are as expected because the value of *Ease of communication* increases when the *Registrar* gets more responsibilities and the *Professor* and the *Student* have first to communicate with him to access the system. On the other hand, for *Process agility* we consider that software actors are more agile than human actors and, thus, its value decreases as the *Professor* and the *Student* depends on the *Registrar*, because it is an intermediate human actor and, so, makes the process less agile. Therefore, despite the three alternatives score the same functional size, they provide different non-functional properties. We remark that, although this is not show for the example, alternatives showing different ways of defining the Functional Processes could lead to different results of CFP.

**Table 2.** Evaluation of the Alternatives

Alternative	CFP	Ease of Communication	Process Agility
Alternative O	107	0.2824	0.7903
Alternative A	107	0.2796	0.8040
Alternative B	107	0.3761	0.6351

## 6 Related Work

There are other Functional Size Measurement Methods that aims at determining the size of a proposed software system yet to be built based on its requirements. Most of these methods are based or are variations of Function Point Analysis (FPA) [3], which is currently being managed by the International Function Point Users Group (IFPUG) [15]. Among the advantages of using the COSMIC method in PRiM instead of FPA or its variations, we remark two. On the one hand, FPA requires weighting the structural elements measured, which introduces a qualitative factor into the process that is not present in COSMIC. On the other hand, COSMIC only distinguishes four different types of data movements that can be easily represented in  $i^*$ , whilst FPA takes into account other aspects such as the number of user inquiries or number of external interfaces, which are more difficult to differentiate in  $i^*$  models.

About related work using COSMIC, [4] instantiates COSMIC by defining a measurement protocol, which describes a set of rules to map the OO-Method (an automatic software production method) requirements model primitives onto the COSMIC concepts. The proposed measurement method is validated in [5]. The work presented in [13] intends to avoid model reconstruction dedicated only to measuring purposes and, in order to address this issue, it proposes a use case model style and detailed measurement principles that intend to achieve synergy between requirement specification and size estimation. Following the same principle, existing work addresses functional size measurement using a modelling approach other than Functional User Requirements, for instance, UML Diagrams [21], [14], or the software model of the process [22].

About the evaluation of non-functional requirements, [19] uses a softgoal interdependency graph in order to calculate the functional size of the non-functional requirements. The proposal also addresses the generation of test cases for non-functional requirements verification purposes. The generation of test cases for testing purposes is addressed in [2], which generates them by combining the functions measured by the COSMIC method with a black box testing strategy. In order to assign priorities to test cases it proposes a functional complexity measure based on entropy measurement.

## 7 Conclusions and Future Work

PRiM is a process reengineering method that aims at assessing non-functional properties of the system using an  $i^*$  requirements model. In order to provide PRiM with functional size measurement capabilities, we have adapted the COSMIC measurement process model to PRiM, and we have checked that the measurement results are correct by replicating existing case studies. We have also used these case studies to generate alternatives and evaluate their non-functional properties with the structural metrics proposed in PRiM.

Based on the results obtained so far, we argue that both methods benefit from this process. On the one hand, the COSMIC method provides PRiM with a standardized measurement process for evaluating the functional size, that has been already validated and it is currently used in a wide variety of situations and domains [6]. Because of that, the COSMIC method also provides knowledge and experience for calculating the functional size. For instance, the questions for validating that a candidate process is a COSMIC functional process [1], or the guidelines for the identification of the entry data movement type provided in [5].

On the other hand, PRiM provides the COSMIC method with the possibility of using a unique  $i^*$  requirements model for representing both functional and non-functional requirements, as well as techniques for gathering the requirements. It also provides guidelines for generating alternatives and structural metrics for evaluating non-functional properties. All these activities can be undertaken by using  $i^*$  models, which avoid having more than one representation, and can be integrated into the development process because it is possible to generate the use cases specification of the information system from the resulting  $i^*$  model.

According to the usability of our proposal, the *i\** classification that we present, indicating which actor is a Functional User, belongs to the Functional Process or represents the Persistent Storage, makes implicit which dependencies are an entry, an exit, a read or a write, facilitating the implicit classification of the Data Groups. Finally, as we have included the COSMIC method in the tool J-PRiM, we already have tool-support for generating and evaluating the models, making possible to define variations over the COSMIC method evaluation formula with little effort.

As future work we will address other metrics based on the functional size, such as the COSMIC functional size of changes [1], or the ones for assessing software product lines, which will provide a better assessment of the alternatives generated with PRiM. As Data Groups are often obtained from class diagrams, we plan to study how to get the class diagrams in addition to the use cases in PRiM.

**Acknowledgements.** The author wants to thank the IWSM-MENSURA 2007 reviewers and assistants for their comments on the earlier version of this paper [11] and, specially, to Charles Symons, who has reviewed the COSMIC terminology used in [11] to adhere to COSMIC v3.0. The author also thanks Xavier Franch for his valuable suggestions. This work has been supported by an UPC research scholarship.

## References

1. Abran, A., et al.: COSMIC Method Version 3.0, Measurement Manual. The Common Software Measurement International Consortium (2007) Last visited: January 2007, <http://www.gelog.etsmtl.ca/cosmic-ffp/COSMIC-MethodV3.html>
2. Abu Talib, M., Ormandjieva, O., Abran, A., Khelifi, A., Buglione, L.: Scenario-based Black-Box Testing in COSMIC-FFP: A Case Study. In Software Quality Professional - Journal of the American Society for Quality 8(3), 22–33 (2006)
3. Albrecht, A.J., Gaffney, J.E.: Software Functions, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. IEEE Transactions on Software Engineering 9(6), 639–647 (1983)
4. Condori-Fernández, N., Abrahão, S., Pastor, O.: Towards a Functional Size Measure for Object-Oriented Systems from Requirements Specifications. In: Proceedings of the 4th International Conference on Quality Software, QSIC 2004, pp. 94–101 (2004)
5. Condori-Fernández, N., Pastor, O.: Evaluating the Productivity and Reproducibility of a Measurement Procedure. In: Roddick, J.F., Benjamins, V.R., Si-said Cherfi, S., Chiang, R., Claramunt, C., Elmasri, R.A., Grandi, F., Han, H., Hepp, M., Lytras, M., Mišić, V.B., Poels, G., Song, I.-Y., Trujillo, J., Vangenot, C. (eds.) ER Workshops 2006. LNCS, vol. 4231, pp. 352–361. Springer, Heidelberg (2006)
6. The COSMIC-FFP at Last visited: January 2007, <http://www.cosmicon.com>
7. The COSMIC-FFP at Last visited: January 2007, <http://www.lrgl.uqam.ca/cosmic-ffp/>
8. Franch, X., Grau, G., Quer, C.: A Framework for the Definition of Metrics for Actor-Dependency Models. In: Proceedings of the 12<sup>th</sup> IEEE International Conference on Requirements Engineering, RE 2004, pp. 348–349 (2004)
9. Grau, G., Franch, X., Maiden, N.A.M.: PRiM: an *i\**-based process reengineering method for information systems specification. Information and Software Technology 50(1-2), 76–100 (2008)



10. Grau, G., Franch, X.: Reef: Defining a Customizable Reengineering Framework. In: Krogstie, J., Opdahl, A., Sindre, G. (eds.) CAiSE 2007 and WES 2007. LNCS, vol. 4495, pp. 485–500. Springer, Heidelberg (2007)
11. Grau, G., Franch, X.: Using the PRiM method to Evaluate Requirements Models with COSMIC-FFP. In: Proceedings of the International Conference on Software Process and Product Measurement, IWSM-Mensura 2007, pp. 110–120 (2007)
12. Grau, G., Franch, X., Ávila, S.: J-PRiM: A Java Tool for a Process Reengineering *i*\* Methodology. In: Proceedings of the 12<sup>th</sup> IEEE International Conference on Requirements Engineering, RE 2006, pp. 352–353 (2006)
13. Habela, P., Glowacki, E., Serafinski, T., Subieta, K.: Adapting Use Case Model for COSMIC-FFP Based Measurement. In: Proceedings of the 15th International Workshop on Software Measurement, IWSM 2005, pp. 195–207 (2005)
14. Harput, V., Kaindl, H., Kramer, S.: Extending Function Point Analysis to Object-Oriented Requirements Specifications. In: Proceedings of the 11th International Metrics Symposium, METRICS 2005 (2005)
15. ISO/IEC 19761: Software Engineering – COSMIC-FFP – A functional size measurement method, International Organization for standardization, 2203 (2003)
16. International Function Point Users Group, at Last visited: January 2007, <http://www.ifpug.org/>
17. Jacquet, J.P., Abran, A.: From Software Metrics to Software Measurement Methods: A Process Model. In: Proceedings of the 3rd International Software Engineering Standards Symposium, ISESS 1997, pp. 128–135 (1997)
18. Jones, S., Maiden, N.A.M., Manning, S., Greenwood, J.: Human Activity Modelling in the Specification of Operational Requirements: Work in Progress. In: Proceedings of the Workshop Bridging the Gaps between Software Engineering and Human-Computer Interaction (2004)
19. Kassab, M., Ormandjieva, O., Daneva, M., Abran, A.: Size Measurement of Non-Functional Requirements and their Testing with COSMIC-FFP. In: Proceedings of IWSM-Mensura 2007, pp. 247–259 (2007)
20. 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) Last visited January 2007, <http://www.gelog.etsmtl.ca/cosmic-ffp/casestudies/>
21. Poels, G.: Definition and Validation of a COSMIC-FFP Functional Size Measure for Object-Oriented Systems. In: Proceedings of QAOOSE 2003 (2003)
22. Santillo, L., Conte, M., Meli, R.: Early & Quick Function Point: Sizing More with Less. In: Proceedings of the 11th IEEE International Software Metrics Symposium, METRICS 2005 (2005)
23. Yu, E.: Modelling Strategic Relationships for Process Reengineering. PhD. thesis, University of Toronto (1995)

# Implementing Software Measurement Programs in Non Mature Small Settings

María Díaz-Ley<sup>1</sup>, Félix García<sup>2</sup>, and Mario Piattini<sup>2</sup>

<sup>1</sup> Sistemas Técnicos de Loterías del Estado (STL)  
Gaming Systems Development Department 28234 Madrid, Spain  
Maria.diaz@stl.es

<sup>2</sup> University of Castilla-La Mancha  
Alarcos Research Group – Institute of Information Technologies & Systems  
Dep. of Information Technologies & Systems – Escuela Superior de Informática  
{Felix.Garcia,Mario.Piattini}@uclm.es

**Abstract.** Although measurement has been successfully applied in various areas, it has proved to be a complex and difficult undertaking in the field of software and especially in the context of small and medium enterprises (SMEs). Measurement programs in SMEs with a low maturity level should be tailored to their limitations (limited resources, experts, etc.) in order to carry out the measurement initiatives efficiently. In this paper we report the method, principles and practices followed in our experience for defining and implementing a measurement program in the development department of Sistemas Técnicos de Loterías del Estado (STL). We also show the characteristics of this company which guided our approach towards tackling the problem, and the resulting software measurement program. As a result of the application of certain practices in the company, some significant benefits were obtained, which could be replicated in similar environments.

**Keywords:** Software measurement program, small and medium settings, case study.

## 1 Introduction

A software measurement program is the result of an initiative meant to define and implement the whole process required to obtain and treat certain software information needs. A successful measurement program is the one that becomes a good tool [1], this means that it directly contributes to solving a part of the engineering problem at hand and generates value rather than data [2].

Although measurement is applied in various areas, it has proved to be a complex and difficult undertaking in the field of software, especially within the context of small and medium enterprises [3].

Small organizational units are just as likely to be confronted with demands for credible evidence about their ability to deliver quality products on time and on budget as large, multinational organizations. Similarly, managers in small settings are equally or even more likely than their counterparts in larger organizational units to have to make well-founded business decisions about process improvement and technology

adoption, and must have the wisdom of taking new business opportunities. Therefore, implementing serious measurement programs is even more important in small organizational settings [4].

Unfortunately, small and medium enterprises have some common characteristics which eventually become obstacles in establishing software measurement initiatives. Some of these are: limited resources and training, poor software measurement knowledge, restricted cash flow, a restricted mentality as regards software measurement, etc. [5-7].

This paper aims to report our experience in setting up a measurement program in the software development department of a medium-sized company with a low software measurement maturity level. We show the methodological approach and implementation strategy chosen which fits the characteristics of the company, and we briefly present the resulting measurement program. Finally it is exposed the benefits of using these practices in similar settings.

The paper is organized as follows: Section 2 sets this work in context by showing some measurement program implementation experiences. Section 3 specifies the characteristics of the company, the measurement program definition framework chosen, the organizational approach, the goals of the measurement program, the resulting measurement program and some practices for implementing the measurement program are also expounded. Section 4 shows the conclusions and lessons learned from the experience and Section 5 shows further research.

## 2 Related Work

In this section we provide an overview of the implementation of measurement programs in software technological companies and we address some studies which identifies good practices for implementing software measurement programs successfully.

Daskalantonakis presented a company-wide software measurement initiative which was implemented to fulfill organizational quality policy [8]. In this work some cultural issues were identified and an organizational approach through which to carry out measurement programs was defined. Some benefits were an improvement in terms of defect density and customer satisfaction. In [9] a measurement program at Daniro J-Technologies in the context of software development projects is described where a measurement program definition primary focussed on understanding productivity and defects for the organization was proposed. However the measurement program shown is only in its initial (i.e. planning and definition) stage and the implementation and validation phase is yet to be done.. In [10] Kilpi shows how Nokia organizationally carries out its measurement activities. It explains the differences as regards GQM method which basically are: the use of quality metrics library instead of defining a new set of metrics for each project, the use of a Quality Plan and Metrics Guidelines instead of a GQM plan, the automation of data collection and reporting, and part-time instead of full-time people involved.

With regard to the implementation of measurement programs in small and medium settings, in [11] MacDonell et al. present a measurement program definition whose purpose was to define a metrics framework which serve to determine the development effort in organizations that develops multimedia systems. Lavazza et al. presented a measurement program experience which took place in the Banca Conboto where

GQM was used and adapted due to the operational, budget and time constraints. The resulting measurement program gave valuable information about the quality of the maintenance process [12].

These experiences gave us a valuable insight into the matter but there is very little information about experiences related to small and medium settings in defining and implementing measurement programs.

- 
1. **Incremental Implementation**
  2. **Well-planned metrics framework**
  3. **Use of existing metrics materials**
  4. **Involvement of developers during implementation**
  5. **Measurement process transparent to developers**
  6. **Usefulness of metrics data**
  7. **feedback to developers**
  8. **Ensure that data is seen to have integrity**
  9. **Measurement data is used and seen to be used**
  10. **Commitment from project managers secured**
  11. **Use automated data collection tools**
  12. **Constantly improving the measurement program.**
  13. **Internal metrics champions used to manage the program**
  14. **Use of external metrics gurus**
  15. **Provision of training from practitioners**
- 

**Fig. 1.** Hall and Fenton measurement success factors [13]

In the frame of measurement programs good practices, Gopal et al. [14] identified and proved some success factors by analyzing its effects on the measurement programs success. The success of a measurement program was measured using two variables: use of metrics in decision-making and improved organizational performance. The success factors selected were divided in two sets: organizational and technical factors. Daskalantonakis also stated some good practices from its experience in Motorola [8, 15], and Fenton and Hall [13] identified from the experience fifteen success factors for implementing software measurement programs as shown in figure 1. However none of these studies show good practices to follow especially when the measurement program is implemented in small and medium software companies and its characteristics are typical of these environments: low measurement maturity level, poor measurement knowledge, measurement not integrated in the culture, limited resources and budget, etc. In figure 1 the success factors detected by Fenton and Hall that do not easily fit for SMEs are underlined.

### 3 Development of the Measurement Programs in STL

In this section we describe the company in which the case study was conducted, the methodological and organizational approach set out, the principles followed; the process improvement goals supported by the measurement program and a summary of the resulting measurement program.

#### 3.1 Description of the Company: Measurement Programs Constraints

Sistemas Técnicos de Loterías del Estado (STL) is a company which was created by the Spanish government and which provides the operations and IT development services for the national lottery. Software measurement initiatives have been encouraged for many years by the software development and maintenance department in this company, which is formed of 39 people. Unfortunately the measurement process which was defined was not accurate enough and had not been properly established throughout the department. Some of the outstanding problems were the scarce amount of resources available for this task. However the need to measure continued and the need of defining accurate measurement programs re-emerged

The development and maintenance department is in charge of developing and maintaining the bet on-line systems including the software of the related terminals and payment channels; and the invoicing and informative systems. They support 33 products of which 21 are critical and 18 of them are on-line systems. Most of the code is written in FORTRAN, C, C++, some Java and PL/SQL. The core critical product amounts to 2000 KLOC.

The development projects were normally carried out by less than 12 people. It usually takes five or six months long and rarely above 15 months.

Some other characteristics of the company related to measurement are as follows:

- C1: The resources were limited and therefore we could not spend too much effort on defining and implementing the measurement program.
- C2: Some project managers were reluctant to use the measurement initiative
- C3: Measurement was not established in the company culture.
- C4: Measurement knowledge was quite limited throughout the company.
- C5: The software measures collected were few, there was no established measurement process and therefore the measurement maturity was quite low.

#### 3.2 Measurement Program Definition Framework

MIS-PyME (Marco metodológico para la definición de Indicadores de Software orientado a PyME) is a methodological framework focussed on defining measurement programs based on software indicators in small and medium settings. The main characteristic of MIS-PyME, among the outstanding software measurement models, is that it fully covers the requirements of a measurement program model suited to small and medium settings with a low measurement maturity level. MIS-PyME framework is classified in three main modules: the methodology and roles (MIS-PyME methodology), the workproducts which give support to the methodology (MIS-PyME Measurement Goals Table, MIS-PyME Indicator Template and MIS-PyME Database) and

the measurement maturity model (MIS-PyME Measurement Maturity Model). MIS-PyME Methodology is based on GQ(I)M [16, 17], but it is designed to define basic indicators which are commonly used and required in most small and medium software development settings. These indicators are adapted to the measurement maturity of the setting. Like GQ(I)M, it is a top-down methodology since it develops the measurement program with the goal in mind but restricts the domain of those goals solely to software process improvement and may be conditioned by the MIS-PyME measurement goals table and the indicator templates provided. This methodology also makes use of a database of measurement program definitions related to software process improvement. MIS-PyME methodology also stresses and helps the integration of the measurement program in the software processes. The principles supported by MIS-PyME are the following:

- The “Reuse and project-specific tailoring” principle, stated by Basili and Rombach [18, 19]. This principle indicates that measurement planning should reuse models and metrics which have been defined for the whole organization. However these models might be tailored to the project or product specific characteristics. MIS-PyME encourages companies in defining measurement programs which will be valid and useful in almost all products or projects, by doing so it makes measurement programs establishment easier in the development unit and make possible the cross-projects and products control.
- Related to the above principle, we base the measurement program on the indicators since they give high level information and can be applied to most of projects, products, processes depending on the nature of the indicator. The indicators allow making cross-analysis and they are an easy reusable unit. Measures and derived measures used in the indicator may be specific for certain project or product, but when then measurement program is to be applied to a new project (or product) these measures or derived measurement of the measurement program will be adapted to the new project.
- To define and implement measurement programs which are adapted to the measurement maturity of the setting. Companies may work in defining and implementing measurement programs that they are able to successfully implement and not to try achieving the best measure when there are several obstacles that make impossible a successful implementation. These obstacles may be related to development, management or quality capability, support tools, etc.
- To define measurement programs with the sole purpose of giving support to the software process improvement initiatives as are also followed in [8]. If the measurement process is still not well integrated in the organization culture and it is not mature enough, it is costly to establish measurement programs. Small and medium settings could not achieve to successfully implement measurement programs which come from just any business goal; at least it will be costly to achieve it. In addition, as stated in the previous point, measurement maturity can not overcome processes maturity. Both areas are extremely connected and need to improve together. Since measurement is not an aim but a way, MIS-PyME leaves the responsibility for seeking a goal to process improvement initiatives.

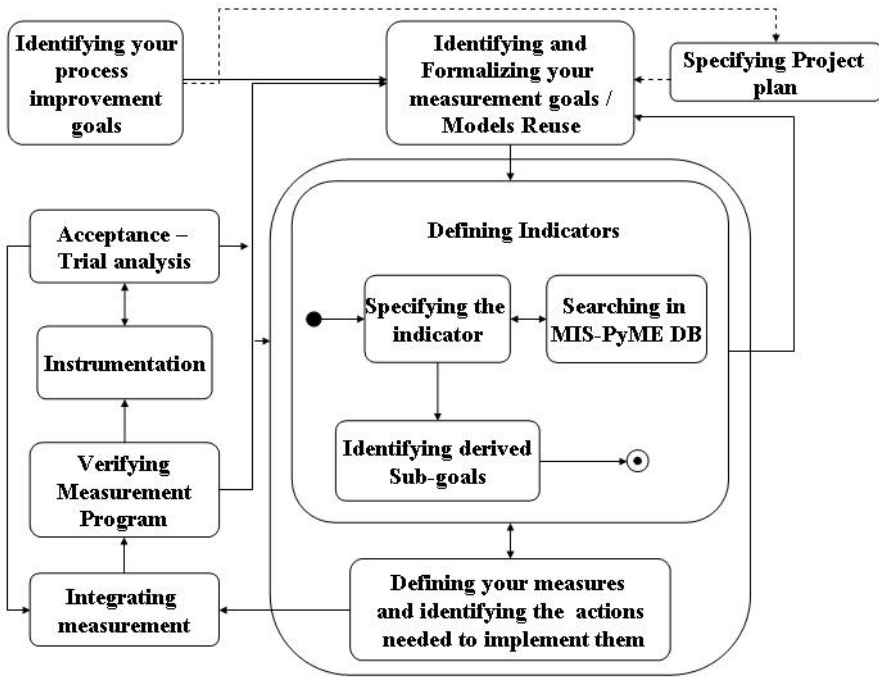


Fig. 2. MIS-PyME Methodology

Figure 2 succinctly outlines this methodology, which is composed of the following activities.

1. **Identifying your Process Improvement Goals:** The process improvement goals that you want to carry out aided by software measurement are defined and the related entities that will help achieve this goal are identified.
  2. **Formalizing Measurement Goals and Check if a Measurement Model is Re-used:** Measurement goals are specified. After that, the object of study, the purpose, the environment and the measurement constraints are defined. MIS-PyME measurement goal table helps the user to identify the measurement goals that will support the process improvement goals. In this step whether there is any measurement goal in the already measurement process of the organization with the same description is also verified. If affirmative, its corresponding indicator template is checked to understand if it fulfils the needs to support the process improvement goal.
- ⇒ **Specifying Project Plan:** A small project plan should be defined. It should only contain: A description of the measurement program, the people involved and their roles, the calendar and a specification of the acceptance phase (trial analysis and pilot project). It may be included in a process improvement project if exists.
3. **Defining Indicators:** Indicators required to implement measurement goals are defined. This is a three step status.

3.1. *Specifying the indicators:* If the measurement goals were in the MIS-PYME measurement goals table, the measurement analyst might take a look at the recommendations, restrictions, questions, etc. according to the MIS-PYME indicator template established for that goal.

3.2. *Searching in MIS-PyME database:* When defining an indicator, measurement analysts may check for any examples in the database related to the MIS-PYME indicator template required for the desired indicator. If a suitable one is found, they can directly and effortlessly adapt the indicator proposed to the measurement program being defined.

3.3. *Identifying sub-goals derived:* Any of the questions posed or the inputs recommended in the MIS-PYME indicator template table may lead to another measurement goal. We call these measurement-derived goals, which may also have their corresponding measurement goal in the table for MIS-PYME measurement goals and their corresponding MIS-PYME indicator templates. Step 2 and 3 may then be repeated until all measurement-derived goals and their relevant indicators have been defined.

4. **Defining your Measures and Identifying the Actions Needed to Implement them:** The measures that have to be collected are identified in detail and defined in the checklists. It is defined which data is to be included/excluded from the measured values, as well as how the data will be collected. The ability of the organization to obtain the measures is analyzed, and the way in which they could be collected is established. If it is not possible to collect the desired data, the indicator specification may be modified based on this information. Maybe it should be defined several ways to get certain measure since this measures can be found in different context (different program languages, different procedures involved, different systems, etc).
5. **Integrating the Measurement Program:** Integrating the measurement activities into previous measurement processes and into other software processes is the aim of this step. MIS-PYME provides guidance in order to integrate the indicators and measurement sub-processes in the development, quality and management sub-processes of the company.
6. **Verifying the Measurement Program:** The measurement process resulting from the process is verified by reviewers and modified if required.
7. **Instrumentation:** Tools which support measurement process are developed or tailored.
8. **Acceptance of the Measurement Program:** The measurement program is used in a trial analysis or in a pilot project.

### 3.3 Settings

The roles necessary to carry out the measurement program and suggested by MIS-PyME model were as follows: the measurement analyst who had some knowledge about software measurement but was not too experienced. The usual work of this role is consulting, coordinating projects, defining requirements and testing. The second role was played by the manager (the director of the development department) who supports the measurement program initiative and has an in-depth knowledge of the software processes and process improvement needs. The third role was the reviewer.



The group of people who played this role was formed of the five project managers (most of them are also the sub-department managers) and some key developers.

This approach was designed in order not to disturb project managers in their usual tasks. The top manager knows the main measurement needs and he is capable of determining them. The measurement analyst will work only with the top manager during the entire definition process phase. At the “verifying measurement process” phase the measurement analyst works with the reviewers.

### 3.4 Software Measurement Program in STL

The highest priority software Process Improvement Goals (PIG) were the following:

- PIG 1: Improving project monitoring and control. We particularly wished to improve the monitoring of the project’s progress in comparison to the plan and to understand and manage any deviations from the plan at the project’s closure. Besides regarding the process, we wanted to monitor the conformance with the test phases.
- PIG 2: Improving the development service and product quality. This goal focused on understanding, monitoring and evaluating the development service and product quality exploited.

**Measurement Program Resulted.** For the first goal (PIG1) three sub-goals through which to improve the software process are identified: PIG 1.1- Improving project progress monitoring in comparison to the plan; PIG 1.2 - Understanding and managing any deviations from the plan at the project’s closure. PIG 1.3: Evaluating the conformance of projects with the test phases.

For the first sub-goal we emphasized the indicators as follows: the first indicator which gives the information about the effort spent against what it was estimated (IND-PRJ-EffortConformance). The next indicator shows the progress of the coding phase by showing the number of requirements coded against the total, and the planned schedule (IND-PRJ-ProgCod). Another indicator shows the progress of the verification phase by showing the number of open incidences and its severity and the planned schedule (IND-PRJ-ProgVerif). For the acceptance phase progress, an indicator shows the number of requirements verified with regard to the total of requirements, the number of open incidences and its severity, the planned schedule, and the defect density (IND-PRJ-ProgAccept).

For the second sub-goal four indicators were defined in order to measure the deviation at the project closure: deviation regarding the effort (Ind-prj-InexacEffort), the size of the software (Ind-prj-InexacSize), the duration of the project (Ind-prj-InexacDuration,) and the total developing cost (ind-prj-InexacCost).

Besides, for each of the indicators related to the PIG 1.2 sub-goal, an indicator was defined in order to perform global and cross-project analyses. All the projects were thus globally analyzed during a six-month period.

For the third sub-goal PIG 1.3 only two indicators were defined, one called IND-PRJ-TestConformance that measured the total failures detected during the integration and acceptance test and the relation of failures detected in both phases and the same for the severe failures type. For this goal an indicator was defined called IND-PRJORG-TestConformance that made a cross-project balance taking into account the type of projects.

From FIG 2, two sub-goals were identified: FIG.2.1 - understanding, monitoring and evaluating the development service provided and FIG.2.2 –understanding and monitoring the quality of the product exploited.

For FIG.2.1 a first level indicator called Ind-prj-QualityDev was defined. This indicator was simply formed from two other indicators, Ind-prj-FiabImpl and Ind-prj-InexacDuration. Ind-prj-FiabImpl aimed to evaluate the reliability of the software developed and released under a project. Ind-prj-InexacDuration aimed to evaluate the deviation in time as regards what was planned. There is also an indicator which globally measures the development process as regards the development service quality. This indicator is called Ind-prjorg-QualityDev.

FIG.2.2 was represented by three indicators which monitor the quality of the products provided by our clients: Ind-prodorg-FiabCrit, Ind-prodorg-FiabSopot, Ind-prodorg-FiabInf. These indicators show the density of failures in production for each product during the period analyzed. Each indicator includes the products related to its product classification: critical, support, informative.

In summary, the resulting measurement program defined 31 indicators, 29 measures, 6 estimations and 9 concept criteria. This measurement program was designed to be used by the whole department. The measurement program was created in two phases which lasted almost three months. FIG.1.1 was tackled in a second phase after FIG.1.2, FIG.2.1 and FIG.2.2 were already implemented and in use.

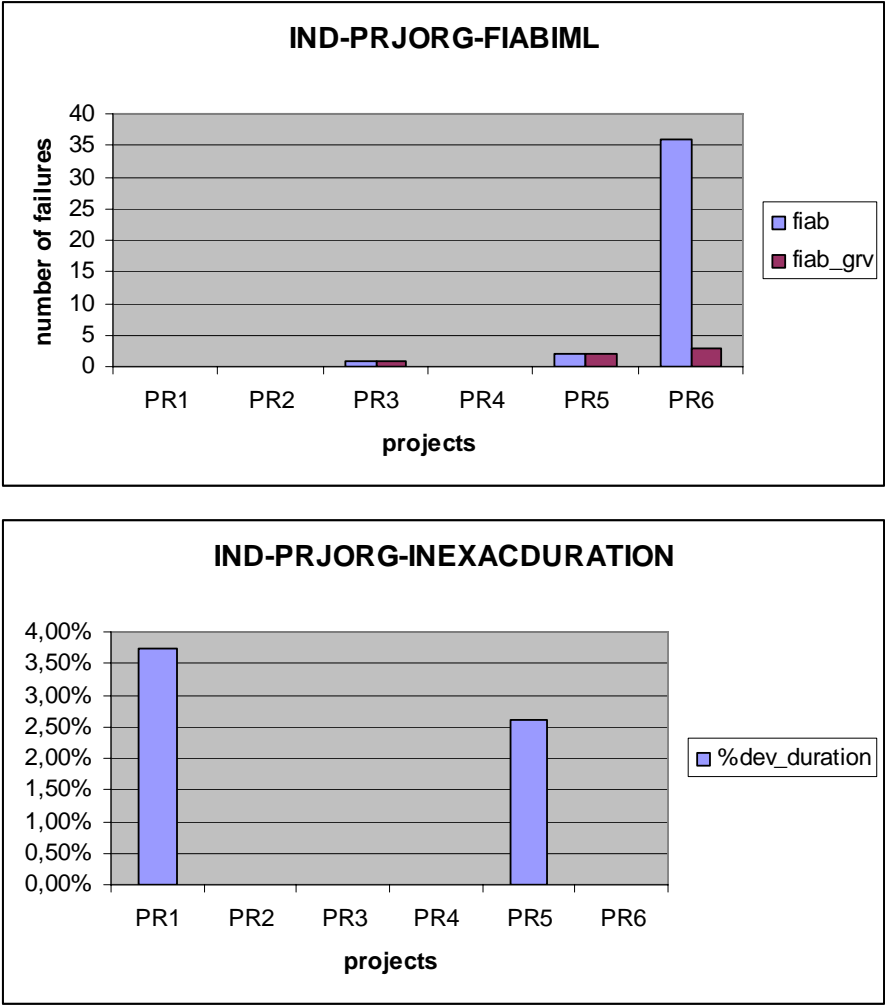
**Software Measurement Process.** The measurement program mentioned above formed the initial measurement process defined for the development department in STL. Three different sub-processes were identified: the project management measurement sub-process, the process management measurement sub-process and the product management measurement sub-process. As far as the integration of the measurement process is concerned, the analysis activities related to the indicators were included in the development process and its related templates (e.g. close of project report, project progress monitoring report, etc. ).

### 3.5 Software Measurement Program Implementation

In this section we briefly explain some important issues related to the implementation of the measurement program and we finally show an example.

**Instrumentation.** It was intended to avoid data that could not be obtained from the tools already in use. The software measurement program collects the information from five development and configuration management tools: the Microsoft Project Manager, ActiTime (a free tool with which to register the effort dedicated to each task), Remedy (an incident management tool) and IRQA (a requirement management tool). Only Remedy and IRQA needed to be tailored in order to automatically launch the queries to the database and to create reports containing the required information and to enable requirement trace. We briefly studied some software measurement tools such as MetricFlame, MetricCenter, ProjectConsole, etc., but we preferred to use simpler tools than complicated ones since the latter may make the understanding of the basic measurement process steps more difficult; and, after weighing up the difference between the effort needed to study a complex tool, adapt it and train people, and the benefits provided, we chose to develop three simple Spread Excel Sheets to give support to each of the sub-processes.

**Verification.** The measurement program was first reviewed in two sessions. In the first session the measurement analyst gave an overview of the measurement program defined and suggestions were only received in the second session after the measurement program was analyzed. Afterwards the measurement program was tested in the department. The results obtained from the measurement programs during the verification phase did not become known outside the reviewers group. For the indicators related to the products, an analysis was done based on the six-month product activity. For the indicators related to FIG 1.2, FIG 1.3 and 2.1, an analysis was done using the projects which were finished between the previous six months. The mentioned



**Fig. 2.** Development service quality indicator (Ind-prjorg-QualityDev) which contains two input indicators: Ind-prjorg-InexacDuration and Ind- prjorg -FiabImpl

indicators were implemented quite fast but PIG1.1 related to the monitoring of the project and therefore its verification took the duration of it. This part of the measurement program is still at the verification phase.

**Final Results and Example.** The measurement program defined in STL finally overcame the goals stated in section 4 (pending the verification phase of PIG 1.1 sub goal). In Figure 2 an example of an indicator (Ind-prjorg-QualityDev) after its first analysis is shown. This indicator not only evaluates the development service quality focused on reliability and project delay for each type of projects, but additionally this indicator made us to ascertain whether the on-time release of the software product had a negative impact on software reliability.

## 4 Conclusions and Lessons Learned

In this paper we have reported an experience in defining and implementing a measurement program in the development department of Sistemas Técnicos de Loterías del Estado (STL) whose aim was to consolidate a useful and simple software measurement process.

The interest of this paper is to show the strategy followed to develop and implement the measurement program, taking into account the characteristics of the company and the good practices identified which could be applied in similar environments.

The characteristics of the company were as follows:

- People involved in the measurement program, including the measurement analyst are from inside the company and not too expertise in the field.
- Poor measurement culture in the company, poor knowledge and therefore poor measurement maturity.
- Some project managers and developers reluctant to use measurement.
- The measurement program should be established in a small or medium software development company or unit with less than 50 people approximately.
- People involved in project, in which the measurement program is established, may be normally less than 12 people.
- The duration of the project should not exceed 15 months and should normally take 5 or 6 months.

Regarding the above characteristics we suggest to follow the practices as follows:

1. The definition of Measurement Programs should not focus on defining measurement programs for certain projects or products, since it will be costly, difficult to handle and of little worth for future developments. The organization should focus on reusing its existing measurement models.
2. Measurement programs should focus on supporting software process improvement goals instead of on business goals. Low measurement maturity level settings cannot afford measurement programs from just any business goal if the aim is to define successful measurement programs effortlessly, accurately and consistently.
3. Measurement Programs definition should be adapted to the measurement maturity of the company. As an example a software measurement low maturity

organization cannot expect empirical prediction results. We used MIS-PyME indicator templates to guide us in this issue.

4. Using common models for software project measurement related may be of use in order to detect the measurement goals which may support software process improvement goals. We made use of the measurement goal table provided by MIS-PyME and we found them useful.
5. With regard to the organizational approach (supported by MIS-PyME) and the roles involved, the benefits detected by our experience were as follows:
  - The extra work that the measurement program definition implies for the project managers is reduced by using this approach. The aim is to seek common useful measurement goals that support software process improvement. The top manager is able to support the measurement analyst in defining the first approach of the measurement program.
  - Project managers make more objective suggestions and effective modifications about the measurement program definition if they review the first approach. As some project managers are reluctant to use these initiatives, the first approach of the measurement program can show them its usefulness and motivate them to review it.
6. We discovered that the way in which the revisions were made by means of two preliminary verification sessions, and the pilot test, was quite useful: By doing so people give better suggestions, analyze better the problem and the usefulness of the measurement program, get more involved in the measurement program, and they easier agree with it.
7. The means of documenting software measurement programs and integrating them in the rest of the software process is essential for the acceptance and use of the measurement process. Our measurement process is documented in a way which clearly answers the questions: "what aspects of the projects, products, resources and process are measured?", "what for?" and "what do I have to measure at each moment of the software development and maintenance process?" Moreover, measurement activities are included in the software development and maintenance model, and in the output report templates involved. Our measurement process was documented in a ".pdf" format but we will, however, modify it to make it accessible via WEB as this is the format of the software development and maintenance model.
8. Excel Spread Sheets or familiar databases are recommended for this type of settings. First because before having powerful tools it is better to understand the process and to control the essential activities. Furthermore, the benefits that the tool provides may not make up the cost of evaluating the tool and training people. Once the company is mature enough, other more powerful tools can be acquired.
9. We also recommend taking advantage of the data provided by already existing development tools and attempting not to collect ambiguous or difficult (as regards data collection) data.

Some of the practices suggested above have been already identified as success factors for implementing measurement programs by some authors: the first recommendation was indicated by Basili and Rombach [19], the second and seventh were identified by Daskalantonakis [8, 15] and the ninth was specified by Fenton and Hall

[13]. We agree with these practices too and we propose others that made the definition and implementation of our measurement program easier.

Since the information related to measurement programs definition and implementation in small and medium settings with similar characteristics is scarce and it covers a wide industry sector, our work is worthy of consideration.

However most of the practices suggested in this study may not be valid when the measurement program is not aimed to be implemented in a software development small and medium company or single unit, when the measurement maturity level is high and the software measurement is integrated in its culture or when the budget and resources assigned are rather good.

## 5 Further Research

Our future research will focus upon guiding small and medium settings towards increasing their maturity with regard to software measurement and also upon observing the benefits of measurement in the development process and its consequences in businesses in this kind of environments. By doing so, we shall continue with the refinement of MIS-PyME methodology framework.

**Acknowledgment.** We would like to thank the staff of Sistemas Técnicos de Loterías del Estado (STL) for their collaboration. This research has been sponsored by the COMPETISOFT (CYTED, 506AC0287), ESFINGE (Dirección General de Investigación del Ministerio de Educación y Ciencia, TIN2006-15175-C05-05) and INGENIO (Junta de Comunidades de Castilla-La Mancha, PAC08-0154-9262) projects.

## References

1. Hughes, R.T.: Expert Judgment as an Estimating Method. *Information and Software Technology*, 67–75 (1996)
2. Niessink, F., Vliet, H.v.: Measurements Should Generate Value, Rather Than Data. In: *Proceedings of the Sixth International Software Metrics Symposium (METRICS 1999)*, Boca Raton (1999)
3. Gresse, C., Punter, T., Anacleto, A.: Software measurement for small and medium enterprises. In: *7th International Conference on Empirical Assessment in Software Engineering (EASE)*, Keele (2003)
4. Goldenson, D., Rout, T., Tuffley, A.: Measuring Performance Results in Small Settings: How do you do it and what matters most? In: *Proceedings of the First International Research Workshop for Process Improvement in Small Settings* (2005)
5. Briand, L.C., Differding, C.M., Rombach, H.D.: Practical Guidelines for Measurement-Based Process Improvement. *Software Process - Improvement and Practice* 2(4), 253–280 (1996)
6. Mondragon, O.A.: Addressing Infrastructure Issues in Very Small Settings. In: *Proceedings of the First International Research Workshop for Process Improvement in Small Settings* (2005)
7. Emam, K.E.: A Multi-Method Evaluation of the Practices of Small Software Projects. In: *Proceedings of the First International Research Workshop for Process Improvement in Small Settings* (2005)

8. Daskalantonakis, M.K.: A Practical View of Software Measurement and Implementation Experiences Within Motorola. *IEEE Transactions on Software Engineering* 18(11), 998–1010 (1992)
9. Kettelerij, R.: Designing A Measurement Programme For Software Development Projects, in Daniro System Integration and Development B.V, May 2006. University of Amsterdam, Amsterdam (2006)
10. Kilpi, T.: Implementing Software Metrics Program at Nokia. *IEEE software* 18(6), 72–76 (2001)
11. MacDonell, S.G., Fletcher, T.: Metric Selection for Effort Assessment in Multimedia Systems Development. In: proceedings of the Fifth International Software Metrics Symposium, Bethesda, MD, USA (1998)
12. Lavazza, L., Mauri, M.: Software Process Measurement in Real World: Dealing with Operating Constraints. In: Workshop on Software Process Simulation and Modeling (2006)
13. Hall, T., Fenton, N.: Implementing Effective Software Metrics Programs. *IEEE software* 14(2), 55–65 (1997)
14. Gopal, A., et al.: Measurement Programs in Software Development: Determinants of Success. *IEEE Transactions on Software Engineering* 28(9), 863–875 (2002)
15. Daskalantonakis, M.K., Yacobellis, R.H., Basili, V.R.: A Method for Assessing Software Measurement Technology. *Quality Engineering*, 27–40 (1990)
16. Park, R.E., Goethert, W.B., Florac, W.A.: Goal-Driven Software Measurement-A Guidebook. Carnegie Mellon University Pittsburgh, Software Engineering Institute (1996)
17. Goethert, W., Sivi, J.: Applications of the Indicator Template for Measurement and Analysis. In: Software Engineering Measurement and Analysis Initiative (September 2004)
18. Basili, V.R., Rombach, D.H.: The Tame Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions of Software Engineering* 14(6), 758–773 (1988)
19. Basili, V.R., Rombach, H.D.: Support for comprehensive reuse. *IEEE Software Engineering Journal*, 758–773 (1988)

# Non-Functional Requirements Size Measurement Method (NFSM) with COSMIC-FFP

Mohamad Kassab, Olga Ormandjieva, Maya Daneva, and Alain Abran

{moh\_kass, ormandj}@cse.concordia.ca,  
m.daneva@utwente.nl, Alain.Abran@etsmtl.ca

**Abstract.** Non-functional requirements (NFRs) of software systems are an important source of uncertainty in effort estimation. Furthermore, quantitatively approaching NFR early in a project is difficult. This paper makes a step towards reducing the impact of uncertainty due to NFRs. It offers a new generic classification of the NFRs, and a NFR size measurement method (NFSM) that incorporates NFRs into the functional size quantification process. We chose the NFR framework as a vehicle to integrate NFRs into the requirements modeling process and to apply quantitative assessment procedures. Our solution proposal also rests on the functional size measurement method, COSMIC-FFP, adopted in 2003 as the ISO/IEC 19761 standard. We discuss the advantages of our approach and the open questions related to its design as well.

## 1 Introduction

Empirical reports consistently indicate that improperly dealing with Non-Functional Requirements (NFRs) leads to project failures, or at least to considerable delays, and, consequently, to significant increases in the final cost as illustrated in the following examples: London Ambulance System (LAS) in 1992 [1], Mars Climate Orbiter in 1998 [2] and Therac 25: The Medical Linear accelerator [28]. According to IEEE software engineering standard 830-1998 [3], NFRs describe not what the software will do, but how it will provide the means to perform functional tasks; for example, software quality attributes, software design constraints and software interface requirements. During requirements elicitation and analysis, NFRs tend to be stated in terms of either the qualities of the functional tasks or the constraints on them, which are expressed as functional requirements (FRs), as the former affect the semantics of the latter.

While estimating development effort is a major activity in managing the scope of the requirements, this activity has, by and large, been neglected for NFRs in practice. This is particularly true because the NFRs are subjective in their nature and they have a broad impact on the system as a whole. Furthermore, NFRs can often be interacting, in that attempts to achieve one NFR can hurt or help the achievement of other NFRs. As NFRs have a global impact on the systems, localized solutions may not suffice [4]. In addition, current software engineering approaches fail in addressing the presentation of how the NFRs can affect several FRs simultaneously. Since this is not supported from the requirements phase to the implementation phase, some of the software engineering principles such as abstraction, localization, modularization,



uniformity and reusability, can be compromised. Furthermore, the resulting system could be more difficult to maintain and evolve.

Despite the above challenges that arise when dealing with NFRs, it is important to be able to make decisions about the scope of software by given resources and budget based on a proper estimation of building both FRs and NFRs. As the effort is a function of size [18], one way to respond to the need to deal comprehensively with the effect of NFRs on the effort of completing the software project is to measure their corresponding functional size. In this paper, the use of the COSMIC-FFP [10, 11] functional size measurement method is proposed to quantify NFR size in a software project. To the best of our knowledge, this is the first attempt to deploy such an approach for NFRs in a project.

The NFR framework outlined in [4] has been chosen to illustrate the application of the proposed measurement method. It was the first framework to propose a process-oriented and qualitative decomposition approach for dealing with NFRs in requirements engineering (RE). A cornerstone of this framework is the concept of the “softgoal”, which is used to represent the NFR. A softgoal is a goal that has no clear-cut definition or criteria to determine whether or not it has been satisfied. In fact, the framework speaks of softgoals being “satisficed” rather than satisfied, to underscore their ad hoc nature, with respect to both their definition and their satisfaction. One drawback of the softgoal approach implied in the NFR framework becomes apparent when we look at solution architecture tradeoffs. The term *softgoal* reflects the fact that extensive interdependencies exist between the various NFRs, and it is often not feasible to entirely fulfill each and every system goal. Tradeoffs must therefore be made. To understand these tradeoffs with respect to the NFR framework, the subgoals are decomposed into *operationalizations*, which provide both candidate design solutions for achieving the goal and the basis for weighing potential tradeoffs. However, the decision-making process for selecting from among different candidate operationalizations which ‘satisfice’ a particular NFR is a qualitative process; that means it typically is not based on defined quantitative criteria. Furthermore, this process is only carried out informally, leaving undocumented the knowledge and the rationale that led to the decisions. This makes it difficult to trace back to the selection criteria on which those decisions were developed.

The above shortcomings underline the need to consider quantitative criteria which make it easier for analysts and software engineers to weigh the various design options and make tradeoffs. In this paper, we address this need by proposing that the softgoal concept in the context of the NFR framework be coupled with NFR size measurement (NFSM). Having the functional size for the softgoal operationalization stated this way provides quantitative criteria for the decision-making task to select the most suitable operationalizations from the candidate alternatives.

The rest of this paper is organized as follows: Section 2 provides a discussion on the types of NFRs. Section 3 presents related work and the NFR framework. Section 4 explains the NFSM method. Section 5 provides a critical discussion of the approach. Section 6 summarizes the key points of the solution proposal and discusses avenues for future research.

## 2 Classification of the NFRs

One of the major challenges when dealing with NFRs is the lack of a formal definition for these requirements; this alone leads to inconsistent interpretational uses of NFRs. In [4], a list of 161 NFRs is presented for the purpose of giving a feel for the breadth of possible areas which can be addressed by NFRs; and yet this is not even close of being a complete list. In many software engineering approaches and industrial practices, NFR is simply used as a blanket term for all requirements that are not explicitly functional. While the concept of a system function is quite sharply defined as a piece of behaviour exhibited by the system, the same cannot be said for NFRs [29].

As of the lack for an accepted definition of NFRs, currently, there is no single universal scheme on how NFRs should be classified. The first attempt to classify NFRs is presented in [30] in which NFRs are considered as software quality characteristics and are presented with a tree-like hierarchy in which meeting a parent quality characteristic implies meeting the offspring sub-characteristics. A more general classification is presented in [35]; and it distinguishes between product, process, resource and project requirements. Other classifications are presented in [32, 33]. Some classifications are offered to certain qualities in particular, e.g. the work of Nielsen on usability [34].

The above discussion highlights the need for a new classification scheme that is compatible with the broad range of possible NFRs that can be suggested from any existing classification. We propose to classify NFRs according to four different criteria: description, satisfaction, association and type. Figure 1 unifies these four classification aspects into a faceted classification of NFRs. Each of the four facets is discussed next.

Description	Satisfaction	Association	TYPE
Stated Implicit	Hard Soft	System Functionality Resource Project Process	Quality Design & implementation constraint Economic constraint Operating constraint Political/cultural constraint

**Fig. 1.** Classification scheme for NFRs

**Description:** For a project under development, not all NFRs are stated in an explicit manner (e.g. written). In the Software Requirements Specifications documents (SRS), many requirements listed as FRs that provide design solutions for NFRs which themselves have never been stated. For example, a requirement could be stated as: “The email messages list in the Inbox should be shown in a descending order based on the time-stamp”. This requirement provides a design solution for the “usability” quality. The usability itself could never be stated. Thus, we classify NFRs according to their description into either stated or implicit. Even when some NFRs are never stated, they

are still there in the stakeholders' minds as constraints on the final product and/or the development process.

**Satisfaction:** In many cases, NFRs are stated as soft requirements with no clear cut definition on how they will be satisfied; e.g. "The gateway shall transfer the credentials in a secure way". The tendency to treat NFRs as softgoals can often add ambiguity to the requirements specifications as in some cases, NFRs could be stated with verifiable terms that should be satisfied in a clear-cut sense. For example, a performance requirement may be stated as: "The gateway must handle 150 transactions per second". This statement represents a hardgoal NFR that is concerned with the quality of the system under development, and, as such, it needs to be absolutely satisfied. This situation calls for distinguishing between two kinds of NFRs based on their satisfaction criteria: soft and hard.

**Association:** Once a software system has been deployed, it is typically straightforward to observe whether or not a certain FR has been met, as the areas of success or failure in their context can be rigidly defined. However, the same is not true for NFRs as these can refer to quantities that can be linked to other elements of the system. In fact, NFRs are not stand-alone goals as their existence is always associated with other concepts.

In this work, we define five types of association points that NFRs and their derived operationalizations, can be associated to throughout the software development process:

- **System Features:** These refer to NFRs that are global in their nature. For example, associating portability, which is a measure of system independence, to the system boundary would intuitively specify that the portability NFR is global and that the system must be operational in multiple platforms, which globally affects every part of the system.
- **Software Functionality (in RUP, a.k.a. to Use Case):** this refers to the context for functionality related NFRs. For example, associating response time NFR to "place-order" use case would indicate that the software must execute the functionality within an acceptable duration. Another example is associating security NFR to the "place-order" functionality, which would indicate that the interaction between Customer and the software system in the "place-order" use case must be secured, which also precisely implies that user interface for other interactions is not required to be secured (e.g. search functionality). We make a note here that a method implementing certain functionality is also considered as "Functionality".
- **Project:** According to [35], A project is defined as the relationship between instances of the following classes of items:
  - A problem to be solved;
  - Internal and external goals and standards;
  - Processes, methods, and techniques;
  - Resources: people, tools and equipment, money, time;
  - Internal and external constraints placed upon the project and/or the use of resources, including those due to the operating environment;
  - A product: one or more defined deliverables.

From the above definition, project-related NFRs specify performance relative to goals, standards and constraints, for example to system access, communication, user/system interface, budget, schedule and scope of the project. Examples of such NFRs are: “The project will follow the Rational Unified Process (RUP)” or “The activities X, Y, Z, will be skipped for this project”.

- **Resource:** Resources are those association points that serve as input to the processes used on a project. Resources include people, tools, materials, methods, time, money and skills [35]. An example of an NFR associated to a resource is illustrated through a requirement like: “The software maintainers have to have 2 years of experience in Oracle database”. This is an operating constraint that is associated with candidates for the maintenance position for the system (another type of resources).
- **Process:** This refers to NFRs that provide precise context for the development process. A process is “a set of activities, methods, practices, techniques and tools used by an organization to develop its products” [5]. Examples of such NFRs are stating staff productivity, stability of the process and process variation (the extend to which instances of the processes differ from each other).

**Type:** Type is a non-functional aspect. The NFR can be of many types. In this paper, five distinguishing types have been identified: Quality, Constraints on design and implementation, operating constraints, political constraints and economic constraints.

- **Quality:** The ISO 9126 [36] is an international standard for the evaluation of software quality. The standard is divided into four parts which address, respectively, the following subjects: quality model; external metrics; internal metrics; and quality in use metrics. The quality model established in the first part of the standard, ISO 9126-1, classifies software quality in a structured set of characteristics as follows: functionality; which represents the set of attributes that bear on the existence of a set of functions and their specified properties, reliability, usability, efficiency, maintainability and portability. Each of these characteristics is further decomposed into sub-characteristics. More in detail the NFRs’ decomposition is discussed in [4].
- **Constraints on the Design and the Implementation:** Constraints are not subject of negotiations and, unlike qualities, are off-limits during design trade-offs. In [31], the constraints on design and the implementation are further decomposed into sub-constraints.
- **Economic Constraint:** including the development cost
- **Political Constraint:** including policy and legal issues
- **Operating Constraint:** including physical constraints, staff availability, etc.

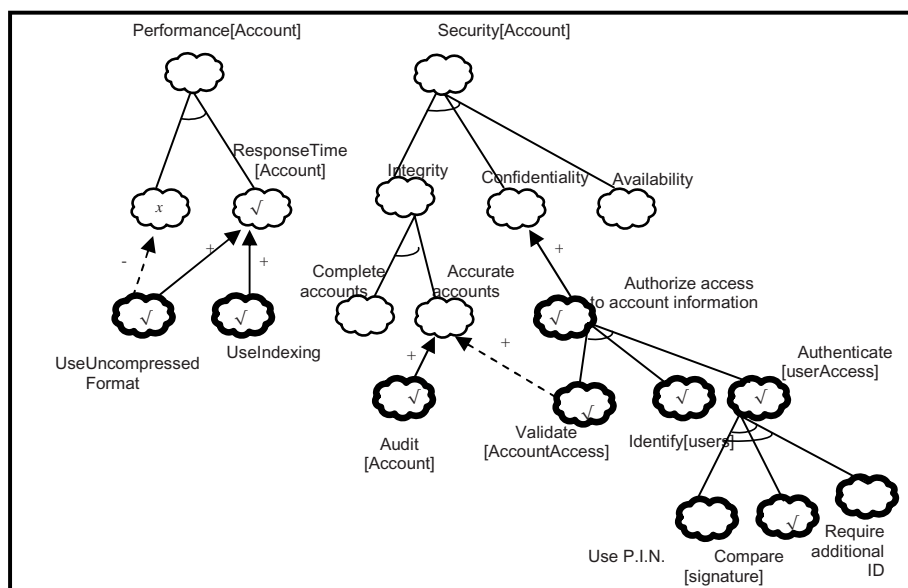
In our approach to measuring the functional size of NFRs, we apply COSMIC-FFP to NFRs that are explicitly stated, hard, regardless of its association or its type. Yet, a further constraint on the proposed NFSM method will be discussed in section 5. More detailed discussion on the constraints on the proposed NFSM method will be presented in section 5.

### 3 NFR Framework and Related Work

The NFR framework [4] is a process-oriented and goal-oriented approach aimed at making NFRs explicit and putting them at the forefront of the stakeholder’s mind. Putting the framework in practice implies executing the following interleaved tasks, which are iterative:

1. Acquiring knowledge about the system's domain, FRs and the particular kinds of NFRs specific to that system of a particular system;
2. Identifying NFRs as NFR softgoals and decomposing them into a finer level;
3. Identifying the possible design alternatives for meeting NFRs in the target system as operationalizing softgoals;
4. Dealing with ambiguities, tradeoffs, priorities and interdependencies among NFRs and operationalizations;
5. Selecting operationalizations;
6. Supporting decisions with a design rationale;
7. Evaluating the impact of operationalization selection decisions on NFR satisfaction.

The operation of the framework can be visualized in terms of the incremental and interactive construction, elaboration, analysis and revision of a softgoal interdependency graph (SIG). Figure 2 presents an example of a SIG with NFR softgoals representing performance and security requirements for customer accounts in a credit card system.



**Fig. 2.** Softgoal interdependency graph for performance and security in a credit card system [4]

In terms of related work, Paech et al. [6] recommend that FRs, NFRs and architecture be tightly co-developed and addressed in a coherent and integrated manner. These authors suggest that NFRs be decomposable into more refined NFRs and additional FRs, as well as architectural decisions. We adopt this proposal, while quantifying the NFRs as described in section 4.

4 NFRs Size Measurement Method (NFSM)

For the purposes of this research, we have chosen to use the functional size measurement method COSMIC-FFP [10] developed by the Common Software Measurement International Consortium (COSMIC) and now adopted as an international standard (ISO/IEC 19761 [11]). Our solution proposal is presented in Figure 3. It shows how the functional size measurement of NFRs is integrated into the NFR framework. We see the NFR framework as the vehicle for eliciting, documenting and operationalizing NFRs. We then propose that COSMIC-FFP be applied to obtain the NFR functional size data. These data are then provided to the relevant stakeholders to assist them in their decision-making process.

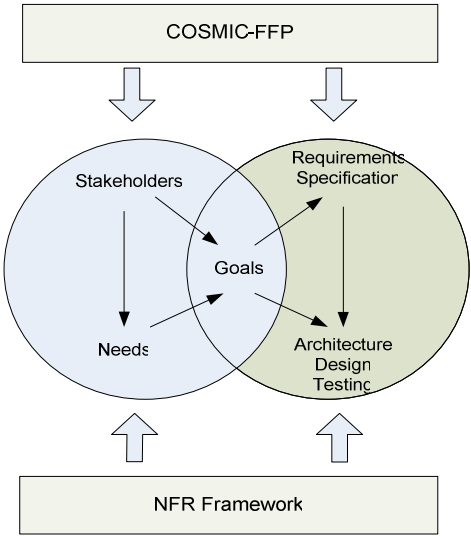


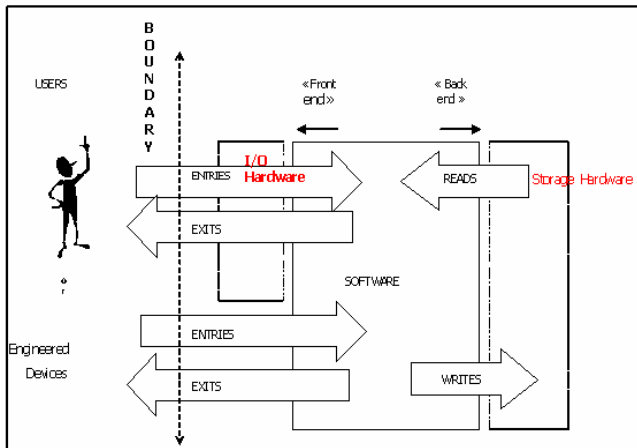
Fig. 3. The solution proposal: a high-level view

4.1 The COSMIC\_FFP Method

The COSMIC-FFP measurement method conforms to all ISO requirements (ISO 14143-1 [12]) for functional size measurement, and addresses some of the major theoretical weaknesses of the earlier Function Point Analysis techniques like Albrecht’s Function Points [13], which dates back almost 30 years to a time when software projects were much smaller and less complex. COSMIC-FFP focuses on the “user view”

of functional requirements, and is applicable throughout the development life cycle, from the requirements phase right through to the implementation and maintenance phases.

The process of measuring software functional size using the COSMIC-FFP method implies that the software functional processes and their triggering events be identified. In COSMIC-FFP, 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. The triggering event is an event occurring outside the boundary of the measured software and initiates one or more functional processes. The subprocesses of each functional process are sequences of events; 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. Figure 4 illustrates the generic flow of data attributes through software from a functional perspective.



**Fig. 4.** Generic flow of data attributes through software from a functional perspective [10]

## 4.2 Size Measurement of NFRs

In our approach, we apply COSMIC-FFP to NFRs stated in verifiable terms (hard). This means that NFRs are stated in terms of crisp indicators with defined acceptable values; thus, it is possible to verify the satisfaction level of those NFRs by comparing the acceptable values with the actual achieved values.

Two views on the size of NFRs are addressed: (1) First, COSMIC-FFP is used to measure the functional size for those operationalizations that correspond to functional processes/functions; and (2) then COSMIC-FFP is used to measure the functional size of the quality control that NFRs require at runtime. NFR quality control is the operation/function that aims to verify the satisfaction of NFRs at runtime (e.g. comparing the acceptable with the actual values).

We state that the size of verifiable NFRs is the sum of both views explained above. The addition of the size values is theoretically valid because COSMIC-FFP size has a unique unit of measurement, the CFP, thus the COSMIC-FFP size measure is at least on the ratio scale. For further discussion on the scale types and the representational theory of measurement, see [27].

**Illustration.** The NFSM method is illustrated on the “availability” NFRs from the Credit Card System example which we borrowed from [4] (see Figure 2).

Two functional processes have been identified in the COSMIC-FFP model for NFR availability, one for each view of the NFR size measurement explained above. The functional processes are: i) availability quantification operationalization, for sizing the reliability of the availability; and ii) availability monitoring, for sizing its monitoring at runtime. Our assumption here is that the availability measurement model is time-dependent, and thus requires a record of the history of system failures, which is stored by the Credit Card System. Moreover, each change in the failure history triggers an update of the availability level, which is modeled as the Availability Quantification process in Table 1. At the same time, the Credit Card System can request a report on the current availability level. This request triggers the Availability Monitoring process (see Table 1), which analyzes the current availability level and issues an “acceptable level” or “critical level” report, depending on the analysis results.

The COSMIC-FFP functional processes and their functional size calculation are illustrated in Table 1. (In this table, the heading of the sixth column, DMT, stands for Data Movement Type.) The total COSMIC-FFP functional size of the availability operationalization and monitoring is 7 CFP.

Similarly, we perform this measurement for all non-decomposable operationalizations that correspond to functional operations/functions. Calculating the functional size of the NFRs is a bottom-up measuring process, in which the selected operationalizations are aggregated to calculate the sub-NFRs and the respective NFRs until the final value is obtained. The functional size of the control functions is added during this process when applicable (see Figure 5 for an illustration of the process).

This task should be performed immediately following task 3 and prior to task 4 in the NFR framework process presented in section 3. The measurement data will provide the rationale required for selecting the appropriate operationalizations. For example, the two operationalizations “Compare Signature” and “Use P.I.N.” are 3 CFP and 2 CFP in size respectively; we have to choose one operationalization to satisfy “Authenticate”, and then we may consider choosing “Use P.I.N.”, as it has a smaller functional size and will thus require less effort/cost to be implemented.

At the same time, some of the NFRs, such as availability, require continuous control (quantification and analysis of the measurement data) at runtime to obtain



Table 1. COSMIC-FFP Data Movements

Process ID	Process description	Triggering event	Data movement identification	Data Group	DMT	CFP
1.1	Availability Quantification	New Failure Data Signal	Receive triggering event	New Failure Data Signal	E	1
			Read Failure History	Failure History	R	1
			Write Current Availability Level	Current Availability	W	1
Functional size in CFP = $\Sigma$ CFP						3
1.2	Availability Monitoring	Monitor Availability Signal	Receive triggering event	Monitor Availability Signal	E	1
			Read Target Availability Level	Target Availability	R	1
			Read Current Availability Level	Current Availability	R	1
			Send acceptable/critical level message	Report	X	1
Functional size in CFP = $\Sigma$ CFP						4
Total COSMIC-FFP points in CFP = $\Sigma$ CFP						7

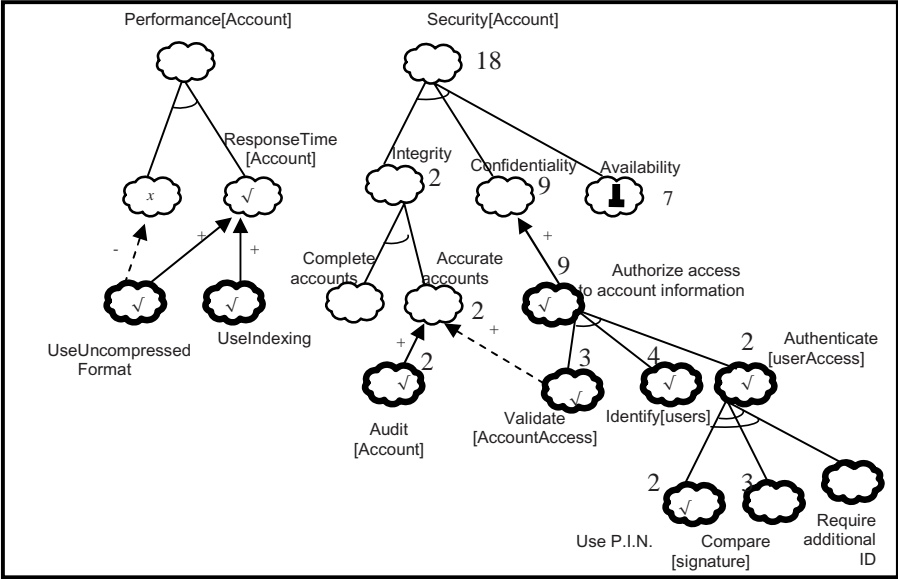
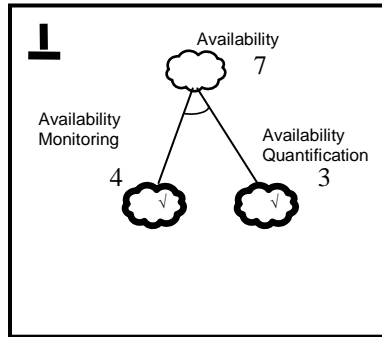


Fig. 5. Calculating functional size for NFRs

feedback on the overall quality of the application. We consider monitoring the quality NFR as an NFR subprocess attached to the corresponding softgoal, and introduce a special symbol to denote such a subprocess:





**Fig. 6.** Refinement of the Availability control subprocess

The subprocesses are further refined into Quantification and Monitoring operationalizations, the size of which is measured with the COSMIC-FFP method in case these subprocesses correspond to functions (see, for instance, Table 1). This approach is illustrated on the Availability NFR (see Figures 5 and 6).

## 5 Discussion

While our proposed NFSM method to measuring the size of NFR makes sense and sounds intuitive, it is far from being issue-free or straightforward to apply. If its purpose is to provide estimators with more realistic size and effort estimates, then we have to make a fine distinction between the two directions estimators may take in quantifying NFRs.

Some publications [22, 23] suggest that, in order for estimators to be able to obtain size and effort numbers for the NFRs in a project, the NFRs must be first decomposed into a series of corresponding FRs. Once this has been done, a functional size measurement method is considered to be the suitable vehicle for quantifying the contribution of NFRs to software size, and, ultimately, to the effort it would take to build the software project. In these publications, it is assumed that it makes sense to decompose all NFRs into FRs. Recently, however, this assumption has become a subject of discussion among some RE researchers [14, 15, 16], who support the position that not all NFRs should be decomposed into FRs. Clearly, there is agreement in the literature that the majority of NFRs can and should be decomposed into FRs, but these RE researchers maintain that there are specific types of NFRs which cannot be decomposed into FRs. Specifically, the goal-oriented RE community [14,15,16] considers that NFRs should not be decomposed into FRs if: (i) the NFRs are normative, that is, if they stipulate how the actor in the system environment shall behave when interacting with the system [16]; or (ii) the NFRs serve as criteria for making architectural design choices; that is, the function of these NFRs is to help evaluate alternatives. Examples of such requirements are the statements “*Time zone information shall be kept together with a local timestamp*” [17] and “*For all Quebec users, the system’s language shall be French*”. In a typical requirements document, these NFRs would be stated in textual form and would not be present, for example, in a requirements diagram (e.g. a use case) which documents the business process and data flows that the

system under development must support. Certainly, we can decompose the NFR from *“The system’s language shall be French”* to an FR like *“Each Quebec user is offered the functionality to select a language,”* *“...to select all documents that should use this language,”* *“...to generate reports in this language,”* and so on.

However, it may well make more sense to consider the NFR as a criterion for exploration and for making choices among alternative architecture options. This consideration is motivated by the following observations: (1) the above decomposition into an FR (which is needed for effort estimators and is used for determining size) refers to functionality that the user did not ask explicitly for at the time of RE; (2) the NFR is a norm to which the user and the system must conform [16], in a bilingual environment (such as Canada or Belgium), for example, where the choice of language is not dictated by user request but by corporate standards and national regulations; and (3) the language of an application tells us about the project context, hence it may point to a contextual factor that may well be a source of risk [18] in terms of obtaining realistic size and effort estimates. Moreover, it should be possible for global applications, like ERP-packaged solutions, which typically produce language-specific reports for specific user groups, to prepare reports in the language specified by the user group. The design architects must then choose a way to set up such a multi-language NFR.

Drawing on this analysis of the RE literature, we incorporated John Mylopoulos’ view of NFRs as architectural design selection criteria [14] in our approach to estimating the size and effort associated with NFRs. Our position is also based on the recommendations of software measurement practitioners [18], who maintain that we, the estimators, need to know the project context first and then use our knowledge of that context to arrive at better estimates. It is our understanding, and our position in this paper, that, if the knowledge of the context of how a system will be used is reflected, and captured in the NFRs, then those NFRs that are not decomposable into FRs should be used as criteria for making design decisions. These decisions are made at two levels [18]: at the micro level (for example, how to design a particular module of a system), and at the macro level (for example, which software architecture to employ). Our position also implies that, whenever we make an architectural design decision, we can potentially affect the accuracy of the cost estimates, since making those decisions introduces uncertainty into the estimation process [18]. This issue is aggravated in the RE phase, where we typically have an incomplete picture of the project context. As a result, there is much more uncertainty surrounding the effort required to satisfactorily develop the capabilities to which the NFRs refer. Because we have to judge how significant these uncertainties (due to NFRs) are, we have to account for them in reporting the final project size assessment.

Therefore, we take into account that cost estimation needs are expected to vary based on the nature of the project, a fact which will also be reflected in the NFRs. For example, a brand-new technology, like implementing a cross-business unit-integrated Enterprise Resource Planning (ERP) system [19], the users of which do not know what their NFRs look like, has more sources of uncertainty than an ERP upgrade project. We can reduce these uncertainties significantly by having the NFR measurement activity explicitly included in reporting the total project size measurement value made up of both FR and NFR size. Consequently, we will establish a more precise estimation of the actual development effort of the system. We assume that, based on

particular estimation needs, we may even consider using different sizing methods to address the NFR issue in a specific project.

## 6 Summary and Future Research Plans

This paper reports on an initial solution to deal with the problem of quantitatively assessing the NFR modeling process early in the project. The NFR Framework approach has been chosen to illustrate the integration of NFRs into the requirements modeling process, and for describing NFR quantitative assessment. Our proposal relies on the COSMIC-FFP functional size measurement method.

To the best of our knowledge, the software industry lacks quantitative effort estimation methods for NFRs, and would certainly benefit from the precise and objective size measurement approach proposed in this paper. This is the motivation for four research activities planned for the near future:

- Determine how the size of NFRs impacts the total project cost,
- Conduct case studies to assess the usefulness of the technique (for example, to research what happens when models of operationalized NFRs become larger,
- Derive guidelines for how to systematically deal with those NFRs that can be decomposed into FRs up to a certain level.
- Extending the effort estimation to cover a broader range of the classification scheme proposed in section 2.

## References

1. Finkelstein, A., Dowell, J.: A Comedy of Errors: The London Ambulance Service Case Study. In: Proceedings of the 8th International Workshop on Software Specifications and Design, pp. 2–5 (1996)
2. Breitman, K.K., Leite, J.C.S.P., Finkelstein, A.: The World's Stage: A Survey on Requirements Engineering Using a Real-Life Case Study. *Journal of the Brazilian Computer Society* 1, 13–37 (1999)
3. IEEE Std. 830-1998: IEEE recommended practice for software requirements specifications. *IEEE Transactions on Software Engineering* (1998)
4. Chung, L., Nixon, B.A., Yu, E., Mylopoulos, J.: *Nonfunctional Requirements in Software Engineering*. Kluwer Academic Publishing (2000)
5. Andrew, J.: An Approach to Quantitative Non-Functional Requirements in Software Development. In: Proceedings of the 34th Annual Government Electronics and Information Association Conference (2000)
6. Paech, B., Dutoit, A., Kerkow, D., von Knethen, A.: Functional Requirements, Non-functional Requirements and Architecture Specification Cannot be Separated – A position paper, REFSQ (2002)
7. Moreira, A., Araujo, J., Brito, I.: Crosscutting Quality Attributes for Requirements Engineering. In: 14th International Conference on Software Engineering and Knowledge Engineering, Ischia, Italy, pp. 167–174 (2002)
8. Park, D., Kand, S.: Design Phase Analysis of Software Performance Using Aspect-Oriented Programming. In: 5th Aspect-Oriented Modeling Workshop in Conjunction with UML 2004, Lisbon, Portugal (2004)

9. Adelman, L., Donnell, M.L.: Evaluating Decision Support Systems: A General Framework and Case Study. In: Andriole, S.J. (ed.) *Microcomputer Decision Support Systems: Design, Implementation, and Evaluation*, pp. 285–310. QED Information Science, Wellesley, MA (1986)
10. Abran, A., Desharnais, J.-M., Oigny, 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, Canada, (2003), <http://www.gelog.etsmtl.ca/cosmic-ffp/manual.jsp>
11. ISO/IEC 19761. *Software Engineering: COSMIC-FFP - A functional size measurement method*, International Organization for Standardization – ISO, Geneva (2003)
12. ISO 14143-1.: *Functional size measurement - Definitions of concepts*, International Organization for Standardization – ISO, Geneva (1988)
13. 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)
14. Mylopoulos, J.: *Goal-oriented Requirements Engineering*. In: *Keynote speech at the 14th IEEE International Conference on Requirements Engineering*. IEEE Computer Society Press (2006)
15. Glinz, M.: *Rethinking the Notion of Non-Functional Requirements*. In: *Proc. of the 3rd World Congress for Software Quality, Munich, Germany* (2005)
16. Wieringa, R.: *The Declarative Problem Frame: Designing Systems that Create and Use Norms*. In: *Proc. of the 10th IEEE International Workshop on Software Specification and Design*, pp. 75–85. IEEE Computer Society Press (2000)
17. Wroblewski, M.: *Quality Governance and Production, Software Quality and Service-oriented Architecture*. In: *Proc of 9th International Conference on Quality Engineering in Software Technology, Berlin*, pp. 333–344 (2006)
18. Pfleeger, S.L., Wu, F., Lewis, R.: *Software Cost Estimation and Sizing Methods: Issues and Guidelines*, RAND Corporation (2005)
19. Daneva, M.: *ERP Requirements Engineering Practice: Lessons Learnt*. *IEEE Software* 21(2), 26–33
20. Mylopoulos, J., Chung, L., Nixon, B.: *Representing and Using Nonfunctional Requirements: A process Oriented Approach*. *IEEE Trans. S.E.* 18, 483–497 (1992)
21. Rosa, N.S., Cunha, P.R.F., Justo, G.R.R.: *ProcessNFL: A language for Describing Non-Functional Properties*. In: *Proc. 35th HICSS, IEEE Press* (2002)
22. ISBSG, *Practical Software Estimation*, 2nd edn. International Software Benchmarking Standard Group (2006)
23. FISMA, *Experience Situation Analysis*, Finnish Software Metrics Association (2001), [http://www.fisma.fi/wp-content/uploads/2006/09/fisma\\_situation\\_analysis\\_method\\_nd21.pdf](http://www.fisma.fi/wp-content/uploads/2006/09/fisma_situation_analysis_method_nd21.pdf)
24. Alves, C., Franch, X., Carvalho, J.P., Finkelstein, A.: *Using Goals and Quality Models to Support the Matching Analysis During COTS Selection*. In: *Proc. of the IEEE Int. Conf. on Component-based Systems*, pp. 146–156 (2005)
25. Jureta, I., Faulkner, S., Schobbens, P.-Y.: *A More Expressive Softgoal Conceptualization for Quality Requirements Analysis*. In: *Proc. of IEEE Int. Conf. on Conceptual Modelling (RE 2006)*, pp. 281–295 (2006)
26. Kaiya, H., Osada, A., Kayjiri, K.: *Identifying Stakeholders and Their Preferences about NFR by Comparing Use Case Diagrams of Several Existing Systems*. In: *Proc. of the IEEE Int. Conf. on Requirements Engineering (RE 2004)*, pp. 112–121 (2004)

27. Fenton, N.E., Pfleeger, S.L.: *Software Metrics: A Rigorous and Practical Approach*, 2nd edn. PWS Publishing (1998), revised printing ISBN 0-534-95425-1
28. Leveson, L., Turner, C.S.: *An Investigation of the Therac-25 Accidents*. IEEE Computer 26(7), 18–41 (1993)
29. Alexander, I.: Misuse Cases Help to Elicit Non-Functional Requirements. Computing & Control Engineering Journal 14(1), 40–45 (2003)
30. Boehm, W., Brown, J.R., Lipow, M.: Quantitative Evaluation of Software Quality. In: *Proceeding of the 2nd Int. Conference on Software Engineering*, San Francisco (1976)
31. Non-Functional Requirements Template,  
[http://www.scenarioplus.org.uk/download\\_nfrs.html](http://www.scenarioplus.org.uk/download_nfrs.html)
32. Bowen, T.P., Wigg, G.B., Tsai, J.T.: *Specification of Software Quality Attributes*, A report published by Rome Air Development Center: Air Force Systems Command (1985)
33. Romanm, C.C.: A Taxonomy of Current Issues in Requirements Engineering. IEEE Computer 18(4), 14–23 (1985)
34. Nielsen, J., Mack, R.L. (eds.): *Usability Inspection Methods*. John Wiley & Sons, Inc. (1993)
35. Whitmire, S.: *Object Oriented Design Measurement*. John Wiley & Sons (1997)
36. International Standard ISO/IEC 9126-1: *Software engineering - Product quality - Part 1: Quality model*. ISO/IEC 9126-1:2001 (2001)

# Assessment of Real-Time Software Specifications Quality Using COSMIC-FFP

Manar Abu Talib<sup>1</sup>, Adel Khelifi<sup>2</sup>, Alain Abran<sup>3</sup>, and Olga Ormandjieva<sup>4</sup>

<sup>1</sup> Zayed University, P.O. Box 4783, Abu Dhabi, UAE

<sup>2</sup> ALHOSN University, P.O. Box: 38772, Abu Dhabi, UAE

<sup>3</sup> École de technologie supérieure - Université du Québec, Montréal, Canada

<sup>4</sup> Concordia University, 1455 de Maisonneuve Blvd. W., Montreal, Canada  
manar.talib@zu.ac.ae, a.khelifi@alhosnu.ae,  
alain.abran@etsmtl.ca, ormandj@cse.concordia.ca

**Abstract.** The success of a system development project largely depends on the non ambiguity of its system-level requirements specification document where the requirements are described at the 'system level' and not at the software and hardware level, and which document serves as an input to the design, implementation and testing phases. The way the system requirements specification document is written is sometimes ambiguous from a software viewpoint. This paper approaches the problem faced by the codesign, namely, the ill-defined functionality allocation between the hardware and the software for real time systems and presents an initial solution to it. It discusses what are the system requirements to be assigned to the hardware and what is really to be done by the software? Different decisions can lead then to various alternatives of allocation of functions between hardware and software: this will affect what software will be built and, correspondingly, the final functional size of software while measuring it using COS-MIC-FFP measurement method. This paper presents an initial solution towards understanding the applicability of the COSMIC-FFP functional size measurement method in assessing the hardware-software requirements allocation, and illustrates the approach on a Steam Boiler Controller case study.

**Keywords:** COSMIC-FFP, ISO 19761, system-level requirements specification, codesign, functional size measurement.

## 1 Introduction

Writing system requirements that define unambiguously the hardware-software allocation of the functionality is critical in the system life-cycle. If not detected early, ambiguities can lead to misinterpretations at the time of requirements analysis and specification, or at a later phase of the software development life cycle, causing an escalation in the cost of requirements elicitation and software/hardware development. Detecting ambiguities at an early stage of the system requirements elicitation process can therefore save a great deal aggravation, not to mention cost. The importance of detecting ambiguity earlier in the system development process is also outlined in the

IEEE Standard 830-1998 [1], which describes the practices, recommended by the IEEE for writing an SRS document, and defines the quality characteristics of a “good” SRS document. They are: (1) Correct, (2) Unambiguous, (3) Complete, (4) Consistent, (5) Ranked for importance, (6) Verifiable, (7) Modifiable and (8) Traceable. Here, the definition of “unambiguous” set by the standard corresponds to an SRS document in which each of its statements has only one interpretation. The IEEE standard further mentions that the inherently ambiguous nature of natural language can make the text of an SRS document fail to comply with the above rule, making it ambiguous, and thereby degrading the overall quality of the document.

Even though the documented requirements used for many case studies for real-time systems are coming from known sources such as universities and trusted industrial organizations, there is no information documented about the quality of these requirements.

In the documentation of these available case studies, there is generally no claim that their sets of documented requirements meet some quality criteria such as those specified in IEEE 830. When a requirements case study does not meet such quality of requirement, there may be unclear text or missing details from the specification problem which would impact:

- The developers who would have to implement such requirements later on,
- The measurers who has to measure the software functional size of such requirements.

As a result, different interpretations of the specification problem would lead both to different software to be built as well as to different functional size of such distinct software

In recent work at measuring the software functional size of real-time requirements case studies with unknown degree of quality, it has been necessary to make some assumptions about the specification problem: it was observed that the specification problem in these case studies were described at the ‘system level’ and did not clearly spell out what was to be done by the hardware and what was to be done by the software. The measurers of these case studies had therefore to clarify the ‘software’ requirements while making various assumptions.

The research described in this paper is concerned with the challenges inherent in understanding the initial system textual requirements and assessing the codesign decisions using the functional size measurement. Our hypothesis is that functional size measurement feedback will help the developers in their trade-off analysis when allocating functionality to software and hardware. Our approach is based on the COSMIC-FFP method, which not only helps clarify the allocation while modeling the system functionality but also and provides theoretically valid and thus objective size measurement results; based on the functional size results the effort associated with a given allocation can be further assessed.

The remainder of this paper is organized as follows: section 2 provides background information on the COSMIC-FFP (ISO 19761) measurement method; section 3 introduces the steam boiler case study. Section 4 identifies how requirements at the system level and related different assumptions from unclear text can lead to different functional sizes of desired software. Section 5 explores a solution to overcome the



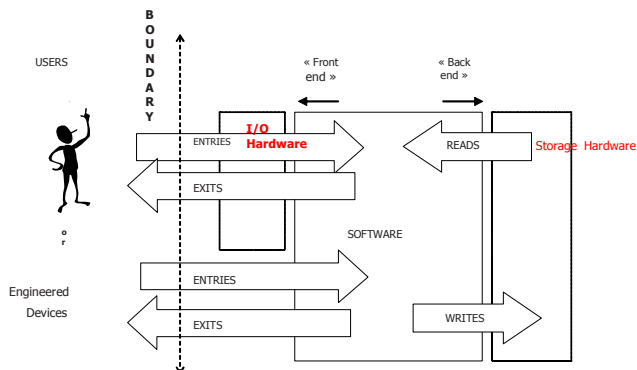
problem of different interpretations of Real-Time Systems Specifications. Finally, section 6 presents a discussion and observations.

## 2 Background

This section introduces key notions of COSMIC-FFP as a functional size measurement method.

This functional size measurement method developed by the Common Software Measurement International Consortium (COSMIC) is an international standard (ISO 19761 [2]) and is referred to as the COSMIC-FFP method [3]. Its design was developed to address some of the major weaknesses of the earlier methods – like FPA [4], the design of which dates back almost 30 years when software was much smaller and much less varied.

In the measurement of software functional size using the COSMIC-FFP method, the software functional processes and their triggering events must be identified. In COSMIC-FFP, the unit of measurement is the data movement, which is a base functional component which 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 measured software and initiates one or more functional processes. The sub processes of each functional process are sequences of events, and a functional process is comprised of 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 1 for an illustration of the generic flow of data groups through software from a functional perspective.



**Fig. 1.** Generic flow of data through software from a functional perspective [2]

### 3 Case Study: Steam Boiler

The Steam Boiler Control specification problem of J. R. Abrial and E. Brger [5] was derived from an original text by J. C. Bauer for the Institute for Risk Re-search at the University of Waterloo, Ontario, Canada. The original text had been submitted as a competition problem to be solved by the participants at the Inter-national Software Safety Symposium organized by the Institute for Risk Re-search. It provides the specification design that will ensure safe operation of a steam boiler by maintaining the ratio of the water level in the boiler and the amount of steam emanating from it with the help of the corresponding measurement devices. The Steam Boiler System consists of the following physical units:

- Steam Boiler: the container holding the water;
- Pump: the device for pouring water into the steam boiler;
- Valve: the mechanism for evacuating water from the steam boiler;
- Water Level Measurement device: a sensor to measure the quantity of water  $q$  (in liters) and inform the system whenever there is a risk of exceeding the minimum or maximum amounts allowed.

Figure 2 shows the Steam Boiler and the relationships between its components. The Steam Boiler is assumed to start up with a safe amount of water. The Controller runs a control cycle every 5 minutes to check on the amount of water currently in the system, and then triggers the Water Level Measurement device and sends the result to the Controller. The Controller receives the current level and checks whether it is normal, above normal or below normal: if the water level is normal, it will do nothing; if there is a risk that the minimum safe level will be reached, the Pump will be triggered to pour more water into the Steam Boiler; and if there is a risk that a level higher than normal will be reached, the Valve will be triggered to evacuate water from the Steam Boiler.

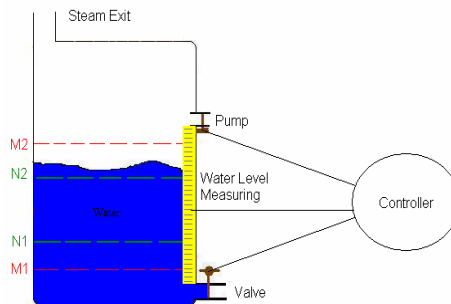


Fig. 2. Steam boiler controller

### 4 Identification of Software Related Ambiguities in the Specifications

The way the specification problem is written is ambiguous from a software viewpoint: at the system level, the specification text talks about a single controller which is the

'system controller'; however, in practice, this system controller consists of two controllers: a hardware part and a software part which are not specified at the system level. The specifications about the interactions between the hardware and the software are sometimes ill-defined in real time applications: what is really to be done by hardware, and what is really to be done by software? For instance, in this case study, the requirements above are at the 'system level', not at the 'software level'. For this case study, a number of alternatives can be proposed of hardware-software allocation, with their corresponding specific requirements - see Table 1.

**Table 1.** Hardware-software allocation alternatives

Alternatives	Hardware controller	Software controller	
1	Generate the five second signal Activate the measuring device Read the output of the measuring device Determine if it is a min, max, normal Send the value (min, max, normal) to the software controller	Receives the current water level signal value (min, max, normal) Based on the values received, it activates the pump or the valve.	
2	Generate the five second signal Activate the measuring device Read the output of the measuring device Send the reading to the software controller	Received the signal value Determine if it is min-max-normal Based on analysis of the (min, max, normal), it activates the pump or the valve.	From the 'system' requirement it is not clear if this min-max is constant, or variable based on some context. If they are constant, this should be clarified in the system-software requirements. If they can vary, then you need some additional 'software requirements' for the ability to manage-update these min-max. You then need a 4 <sup>th</sup> option
3	Received the five second signal to activate the measuring device Read the output of the measuring device Send the reading to the software controller	Generate the five second signal Activate the measuring device Receive the signal value Determine if it is min-max-normal Based on analysis of the min-max-normal, it activates the pump or the valve.	
4	Additional options could be generated: for example, a database that contains the min-max values, and which can be updated by human users.		

For the 'software' view, the text about the controller in the specification problem is ambiguous: for instance, how will the software controller determine whether it is a min or a max?

Alternative 1- The min-max values are specified as constants: then an Entry of these values is needed for the software controller.

Alternative 2- the min-max values are specified as stored values: then a Read of these stored values is needed for the software controller.

Alternative 3- the min-max values are specified as stored updatable values: then an update function is needed (with corresponding data movements).

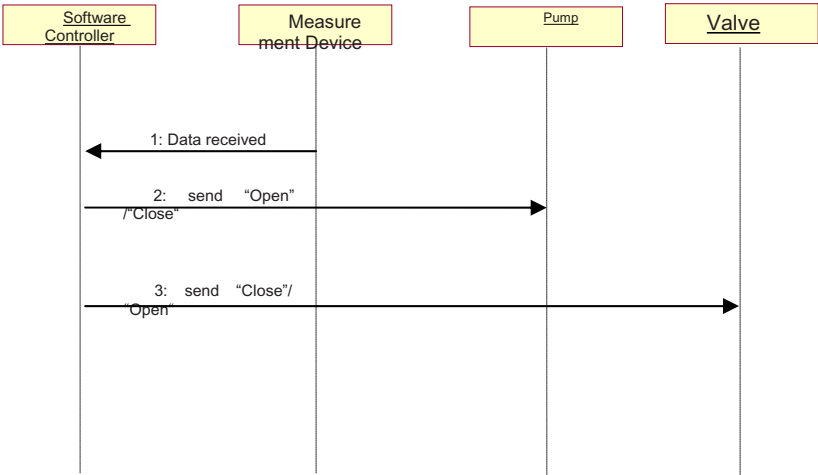
Alternative 4 - additional options could be specified: for example, there could be a requirement for a database that contains the min-max values which can be up-datable by a human operator.

Alternative 1 states that it is the hardware part of the controller that reads the water level, makes the decision (calculation) about the min-max (and the risk of getting close to the min-max), and then sends the outcome (min-max-normal) to the software part of the controller. The software is then only responsible for sending close-open messages to the valve and to the pump.

This alternative 1 then describes the interactions of the software controller with other components, that is, when the water level is below the minimum, normal or above the maximum. This interaction begins when it receives a signal from the hardware measurement device (under the control of the hardware controller every 5 seconds). The basic flow therefore is described as following (see Figure 3):

1. The software receives data from the hardware controller on the water level measurement outcome (min, max or normal).
2. The software analyses the received data. 2. The software obtains the water level measurement outcome (min-max-normal).
3. The software sends an “open”/ “close” message to the pump, which reacts accordingly.
4. The software sends a “close”/ “open” message to the valve.

The sequence diagram for this alternative 1 is presented in Figure 3.

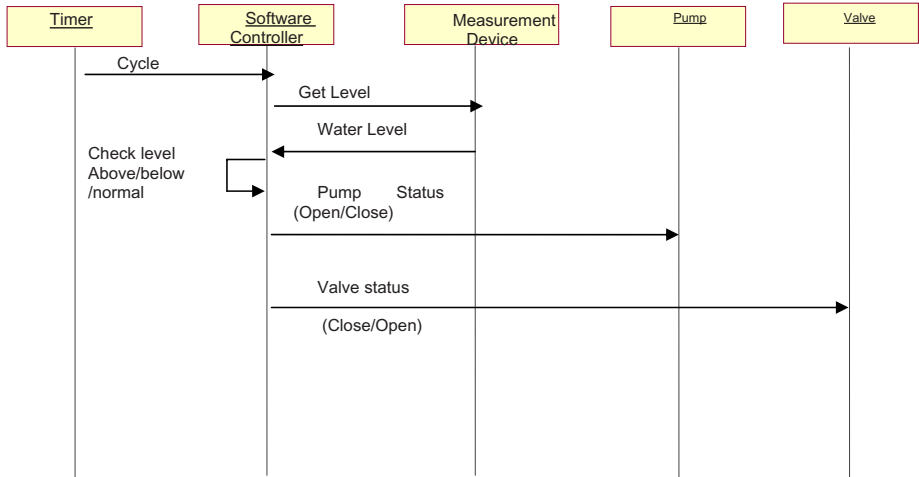


**Fig. 3.** Alternative 1 – Sequence diagram of interactions of the software controller with other components

This alternative 1 would lead to a functional size of 3 Cfsu for the corresponding software – see table 2.

**Table 2.** List of COSMIC-FFP data movements – Alternative 1

Process description	Triggering event	Sub-process Description	Data Group	Data movement Type	Cfsu
Maintain Water Level	Water level signal	Obtain the water level measurement (value = below normal, normal or above normal)	Water level	E	1
		(Logic) Check if any action is needed; if not, terminate the cycle	-		
		Send message to Pump (value = open or close)	Pump status signal	X	1
		Send message to Valve (value = open or close)	Valve status signal	X	1
Total functional size in Cfsu =					3 Cfsu



**Fig. 4.** Alternative 3 – Sequence diagram of interactions of the software controller with other components

Alternative 3, as the next example, states that it is the hardware part of the con-troller that receives the five second signal from the software controller to activate the water level measuring device. Then, the hardware reads the water level and it makes the deci-sion (calculation) about the min-max (and the risk of getting close to the min-max), and then sends the outcome (min-max-normal) to the software part of the controller. The

software is then responsible for generating the five second signal, activating the measuring device and sending close-open messages to the valve and the pump.

The interaction starts when it receives data from the water level measurement device (under the control of the software controller every 5 seconds). Figure 4 shows the basic flow of such interaction as following:

1. The software sends the 5-second signal to software controller.
2. The software requests the current water level.
3. The software obtains the current water level.
4. The software reads the range of the water (Min, Max) and compares the current water level with the Min and Max.
5. The software checks if any action is needed; if not, terminate the cycle.
6. The software sends the new status to the pump (value = open or close).
7. The software sends the new status to the valve (value = open or close).

This alternative 3 would lead to a functional size of 6 Cfsu (COSMIC functional size unit = Cfsu) for the corresponding software – see table 3.

**Table 3.** List of COSMIC-FFP data movements – Alternative 3

Process description	Triggering event	Data Movement Description	Data Group	Data movement Type	Cfsu
Maintain Water Level	5-second signal	Send 5-second signal to Controller	5-second signal	E	1
		Request current water level	Get level signal	X	1
		Obtain the current water level	Water level signal	E	1
		Read the range of the water (Min, Max) and compare the current water level with the Min and Max.	Water Level Range	R	1
		(Logic) Check if any action is needed; if not, terminate the cycle			
		Send new status to Pump (value = open or close)	Pump status signal	X	1
		Send new status to Valve (value = open or close)	Valve status signal	X	1
<b>Total functional size in Cfsu =</b>					<b>6 Cfsu</b>

Similarly, other alternative mixes of hardware-software functions for the same specification problem would lead to different software, each with potentially a different functional size.

It becomes clear, then, that different interpretations can be derived from the same steam boiler specifications description because of missing details regarding the software and hardware requirements.

A generic solution to the above stated problem is presented next.

## 5 A Solution to Overcome the Different Interpretations of the Specification Problem

This section presents steps for a recommended solution derived from [6] for developing a software measurement standard to overcome the different interpretations of the specification problem that lead to different functional sizes of such distinct software. Figure 5 presents the basic steps of such a generic solution as follows:

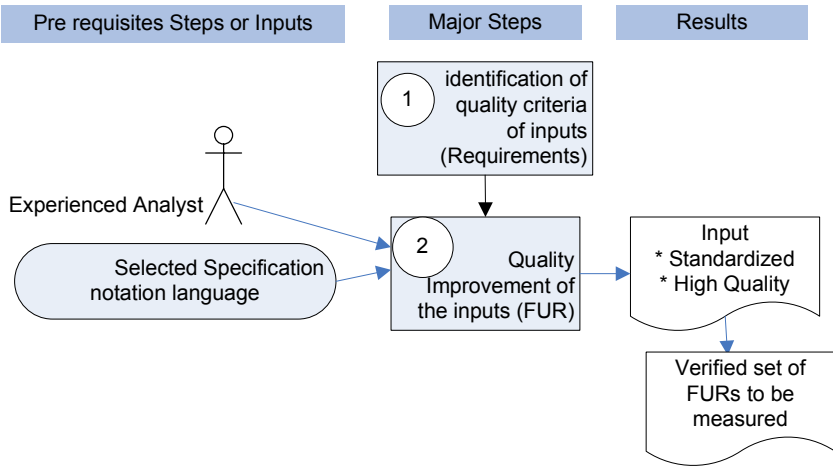
**Step 1.** First, the quality criteria of the functional software requirements (FURs) have to be identified and selected as prerequisites for the input to the measurement process. For the purposes of functional size measurement, the inputs are usually expressed in textual format, and the candidate quality criteria can be found, for instance, in the IEEE standard on Software Specifications Requirements, that is, IEEE 830. These quality criteria then become an input to the second step.

**Step 2.** In this step, the quality of the set of FURs chosen as input to the functional size measurement process is improved by using the quality criteria identified in the Step 1. For this purpose, a specification language is selected and the above set of FURs is translated into the selected specification language. In parallel the FURs are analyzed, verified and improved using the chosen quality criteria (for instance, removing the ambiguities and inconsistencies in the requirements). The output of this step is then the FURs described in the selected notation specification language which meet the specified quality criteria.

To improve the consistency of the documentation to be used as input to the FSM, the UML notation is proposed in this research, such as use cases and sequence diagrams for the software to be measured. The UML Use Case diagram is a tool for representing the entire functionality of a system; a sequence diagram is a structured representation of software behavior as a series of sequential steps over time. Developing such diagrams can improve the comprehension of software functions and provide the measurer with more consistent and precise documentation as input to the measurement process.

The above generic procedure allows the measurer to have his measurement inputs documented in a consistent manner, which in turn allows him greater transparency in the intermediate steps of the measuring process, as well as more repeatable results. For illustrative purposes, Figures 3 & 4 presented two sequence diagrams for the steam boiler controller case study.

An analyst with expertise in UML notation carried out this step, which consisted of analyzing the textual description of the requirements and their transformation into UML notation, and, within this process, correcting defects (for instance, to remove ambiguities and inconsistencies in the requirements). The outcome is the verified set of FURs to be measured.



**Fig. 5.** Generic Steps for Quality Assessment

## 6 Discussion

When a requirements case study is written at the system level, there may be missing details about the allocation of functions between hardware and software both for the developers who will have to implement such requirements later on, as well as immediately for the software measurers attempt to measure the software functional size of such requirements. As a result, different interpretations of specification problem would lead to different functional size of software for the corresponding software.

This paper has therefore explored different alternatives of ‘hardware-software’ requirements from the ‘system requirements’. For the steam boiler application in alternative 1, the total functional size is 3 Cfsu, while with other allocation of functions between hardware and software as stated in alternative 3, this would affect what software will be built and, correspondingly, its final functional size of software, which in this instance was calculated at 6 Cfsu. It can then be observed that different interpretations can be derived from the same steam boiler specification problem because of some missing details regarding the software and hardware requirements.

These findings are significant from three perspectives:

- the writers of such case studies should clearly spelled out that their case studies are documented at the system level, that the hardware-software allocation of functions have not been specified and that the quality of these requirements is not documented and should not be assumed to meet the IEEE-830 quality criteria
- the users of such case studies should beware when they use them for studies related to software functions; the findings observed from the use of the case studies will depend on the specific interpretation of the user of the case studies and might not be generalizable to all potential interpretations of the specific case study, thereby leading to confusion in the mind of the readers, more so if the assumptions about the hardware-software allocation of functions has not been documented. Users of such case studies should therefore verify whether these case studies are documented at the



system level or at a level documenting the hardware-software allocation. Users should also be aware that such case studies, unless specifically documented, that they do not necessarily meet the IEEE-830 quality criteria

- the measurers of such case studies should beware when they use them for studies related to measurement of software functions, and for the same reasons as for the users of these case studies: different allocation of hardware-software functions can lead to different software functional size, and unless the assumptions and interpretations are clearly spelled out and documented during the measurement process, there is little way to justify a specific measurement results and demonstrate that it is the correct size for the specific mix of hardware-software functions. Ultimately distinct measurement results, without detailed documentation of the assumptions used as the basis of measurement, would wrongly lead to lack of confidence in the measurement results themselves, in the measurers ability to come up with the 'correct and repeatable' measurement results and ultimately in the capacity of a measurement method to lead to repeatable measurement results.

A key insights from this study is that the measurement of the software size of a specification problem can bring a number of positive contributions if the measurer clearly document both the ambiguity he has found in a specification document as well as the assumptions he made, such as for the hardware-software allocation of functions, when measuring the functional size of the software.

Therefore, it should be one of the responsibilities of the measurer to identify, within the documented requirements, what he considers as ambiguities and omissions and to document the assumptions he made that led to the measurement results he documented. These comments documented by the measurer represent a value-added contribution to quality during the measurement process, as well as potentially a contribution to reduction of costs later on in the development project. Of course, the measurer's observations and comments should next be re-viewed by the project manager who should then address these comments prior to proceeding further with the project; without such follow-up, there is little improvement to quality.

To overcome the problem of different interpretations of Real-Time Systems Specifications, especially for software developers and software measurers, this paper presented the different steps for assessing and improving the quality of the specifications. This solution is based on suggested basic steps for developing a software measurement standard.

Further work is in progress to analyze additional case studies, verify these observations and derive techniques that would improve both the measurement process and the contributions to the improvement of the quality of the problem specifications.

## References

1. IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications, Software Engineering Standards Committee of the IEEE Computer Society (1998)
2. Abran, A., Desharnais, J.-M., Oigny, S., St-Pierre, D., Symons, C.: COS-MIC FFP – Measurement Manual (COSMIC implementation guide to ISO/IEC 19761:2003). École de technologie supérieure - Université du Québec, Montréal (2003)

3. ISO/IEC 19761. Software Engineering - COSMIC-FFP - A functional size measurement method. International Organization for Standardization - ISO, Geneva (2003)
4. 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)
5. Abrial, J.R.: Steam Boiler Control Specification Problem (1991)
6. Khelifi, A.: A Set of References for Software Measurement with ISO 19761 (COSMIC-FFP): an Exploratory Study, Montréal (Canada). Département de génie logiciel, Ph.D. thesis, École de Technologie Supérieure, p. 495 (2005)

# Analysis of Software Functional Size Databases

Juan J. Cuadrado-Gallego, Miguel Garre, Ricardo J. Rejas,  
and Miguel-Ángel Sicilia

University of Alcalá, Autovia A-2 km 33.6 - 28871,  
Alcalá de Henares, Madrid, Spain  
jjcg@uah.es

**Abstract.** Parametric software effort estimation models rely on the availability of historical project databases from which estimation models are derived. In the case of large project databases with data coming from heterogeneous sources, a single mathematical model cannot properly capture the diverse nature of the projects under consideration. Clustering algorithms can be used to segment the project database, obtaining several segmented models. In this paper, a new tool is presented, Recursive Clustering Tool, which implements the EM algorithm to cluster the projects, and allows use different regression curves to fit the different segmented models. This different approaches will be compared to each other and with respect to the parametric model that is not segmented. The results allows conclude that depending on the arrangement and characteristics of the given clusters, one regression approach or another must be used, and in general, the segmented model improve the unsegmented one.

**Keywords:** Software Engineering, Effort estimation, Segmented parametric model, Recursive Clustering Tool (RCT), Clustering, EM algorithm.

## 1 Introduction

There are several classifications of estimation methods. For example Boehm [13] classifies software cost estimation methods in: algorithmic models, experts's judgment, analogy, parkinson, price to win, top-down and bottom-up. DeMarco [14] and Conte [15] classification is composed of historic-experimental models, statisticians models, theoretic models, and compound models. In 1991 Kitchenham [16] classifies software estimation methods into experts's opinion, analogy, decomposition, and estimation equations, the different methods of estimation. Fairley [17] in 1992 offers this other classification, empiric models, based in regression, and based in the theory. In 1997, Walkerden and Jeffery [18] offer us another classification, empiric parametric models, empiric nonparametric models, analogical models, theoretic models and heuristic models. More recently, Boehm [7] updated his classification comprises: models based techniques, experts based techniques, learning oriented, dynamic models, models based in techniques of regression, and compounds Bayesian models. Within learning oriented methods, Boehm mentions that neural networks and the case study (originated as

from models for analogy). Finally, Wiczcerek [19] classification includes: model based methods and no-model based methods, within the first we have generic (proprietary and not proprietary) and specific (data driven and compounds).

If we focus on learning oriented methods, according to the 2001 classification of Boehm, at present, in addition to neural nets and case based reasoning (CBR) (Shepperd [22]), another machine learning techniques are being applied. Such techniques include Genetic Programming, Fuzzy Systems, Rule Induction, Classification and Regression Trees (CART) and combinations of them. For example, Idri [20,21] applies fuzzy systems combined with analogies. Dolado [23,24] applied genetic programming to estimate software projects cost. In the case of rule induction, Mair [25] compared this technique with neural nets and CBR. Briand [26,27] realized different analysis and comparisons of CART. Lee [28] uses clustering techniques for training of a neural net applied to the software cost estimation.

In this work, we apply clustering techniques to segment a the ISBSG repository; then, different regression models are applied to each cluster. When the projects of a projects database came from a same company, and in the neighborhood of hundred, an only equation was enough for perceiving the dependence between the cost and a linear combination of another independent variables. In the event of databases of thousands of projects, and from different companies and countries, such as the data come from ISBSG database, this is not possible, for so different nature of the projects belonging to it. To do so, if you want accomplishing an analysis of data it is necessary to group previously the projects in groups the more similar possible. This is the idea this paper has based, utilizing in this case, the EM algorithm [5,6,8] to segment the ISBSG database and working at a later time and of independent form with the obtained projects segments.

Segmented parametric software estimation models has been used against not segmented, in a variety of works. Concretely Cuadrado et al [1] show the influence of cost drivers, CASET (Use of Case Tools) and METHO (Use of a Methodology), on estimation effort using segmented parametric models. In a previous work, a recursive process is described to obtain a set of segments that shows more homogeneous characteristics than previously [2], Also we compared the two models (segmented and not segmented) in [3] and in [4], the use of different clustering algorithms to cluster the ISBSG database. These works are based in the use of the EM algorithm, which has been integrated into a software tool: "Recursive Clustering Tool" (RCT). In these works they get a groups set of projects, from the projects database ISBSG version 8 ( International Software Benchmarking Standard Group<sup>1</sup>), and on each their one a mathematical model is applicable  $e = as^b$ , where  $e$  is the effort and  $s$  any measure of the size of the project. Constants  $a$  and  $b$  are obtained by regression analysis on projects of every group. Obtained results (respect to MMRE and PRED (0,3)), show that they improve the ones that they get when utilizing an only equation for all of the projects.

Oligny et al [9] used the ISBSG version 4 to obtain a empirical time-prediction model, based on effort. The relation between time and effort are established

---

<sup>1</sup> <http://www.isbsg.org/>

by means of regression analysis. The most important aspect of this work is that it uses three estimation models. One to the 208 *mainframe* projects, the other to the 65 *mid-range* projects, and the last one to the 39 *pc* projects. The authors distinguish between the different development platforms used to get the model, obtaining a relation for each one of the groups. This study can be considered as supporting evidence for the segmentation approach described in this paper, even though the partitioning of the data is carried out without using a clustering algorithm. Clustering is a suitable technique that has been used to segment the project database in different clusters obtaining more homogeneous characteristics. After that, a set of parametrics models are obtained, one for each cluster. This models are calibrated using curve regression analysis.

The rest of this paper is structured as follows. In Section 2, the data preparation process is shown, Section 3 shows the results for the not segmented model. Section 4 provides the whole process of clustering and the mathematical equations obtained. After this, Section 5 provides the discussion and the empirical evaluation of the approaches carried out in previous section. Finally, conclusions and future research directions are described in Section 6.

## 2 Data Preparation

The entire ISBSG-8 database containing information about 2028 projects was used as the project database. The database contained attributes about size, effort and many other project characteristics. However, before applying clustering techniques to the dataset, there are a number of issues to be taken into consideration dealing with cleaning and data preparation.

An important attribute is the *data quality rating* that can take values from A (where the submission satisfies all criteria for seemingly sound data) to D (where the data has some fundamental shortcomings). According to ISBSG only projects classified as A or B should be used for statistical analysis.

The first cleaning step was that of removing the projects with null or invalid numerical values for the fields effort (“Summary Work Effort” in ISBSG-8) and size (“Function Points”). Then, the projects with “Recording method” for total effort other than “Staff hours” were removed. The reason for this is that the other methods for recording were considered to be subject to subjectivity. For example, “productive time” is a rather difficult magnitude to assess in a organizational context. Since size measurements were considered the main driver of project effort, the database was further cleaned for homogeneity in this respect. Concretely, the projects that used other size estimating method (“Derived count approach”) than IFPUG, NESMA, Albretch or Dreger were removed, since they represent smaller portions of the database. The differences between IFPUG and NESMA methods are considered to have a negligible impact on the results of function point counts [11]. Counts based on Albretch techniques were not removed since in fact IFPUG is a revision of these techniques, similarly, the Dreger method refers to the book [12], which is simply a guide to IFPUG counts.

Another question is the consistency of the attribute values. For example some times the fields appear empty while some times appear with the value “don’t know”. There are another cases in which the used language is COBOL 2 or COBOL II, it is necessary give them the same value, because it is the same language.

After this, 1546 projects remains in the database, which will be used in the next section.

### 3 Single Mathematical Model for the Entire Database

We first show the mathematical equation for the entire project database, and associated the MMRE and PRED(0.3) measures. The equation is  $e = 38.94 \cdot fp^{0.7343}$ . Table 1 show the results obtained using 1246 projects for training and 300 for testing.

**Table 1.** Linear log-log plot regression using 1246 instances for training and 300 for testing

	Seg Avg Pred				Model Pred	
	$A_0$	$A_1$	MMRE	PRED(0.3)	MMRE	PRED(0.3)
Lin log-log	38.94	0.73	1.58	22.44%	1.41	22.34%

The regression used is the linear log-log regression (a double logarithmic transformation of the effort and function points are made, besides it uses least absolute deviation to minimize points deviation).

There are two possibilities to test the models, segmented and not segmented:

1. Get the MMRE and PRED values for the given curves of each one of the clusters. An average value of all this values can be calculated. Option *Build*  $\rightarrow$  *Data Analysis* of RCT.
2. Get the MMRE and PRED values for a set of test data instances. Option *Build*  $\rightarrow$  *Full Test* of RCT.

The first measure give us an idea of the predictive capacity of each one of the curves, and the average values of MMRE and PRED give the predictive capacity of all the curves. The second one give us the predictive capacity of the whole model. Both measures will be used to evaluate the segmented parametric model.

In table 1, the first column “Curves Average Prediction” represents the average value of all the curves MMRE and PRED values. The second column “Model Prediction” shows the MMRE and PRED values obtained predicting the effort of the 300 test projects, this is the second possibility mentioned before.

In the case of not segmented model, the “Curves Average Prediction” have the meaning given before applied to a unique curve.

4 Results of the Clustering Process

The experiment was performed using out RCT tool and consisted in the clustering of 1246 projects selected randomly as a training, the remaining 300 projects in the repository. The attributes considered were *effort* and *function points*, they are supposed to follow a two-dimensional normal distribution.

Table 2 shows the clusters obtained after the first segmentation step using the *effort* and *function points* and related statistics. Figure 1 also shows the graphical representation of these clusters and the test data that will be used in the full test report.

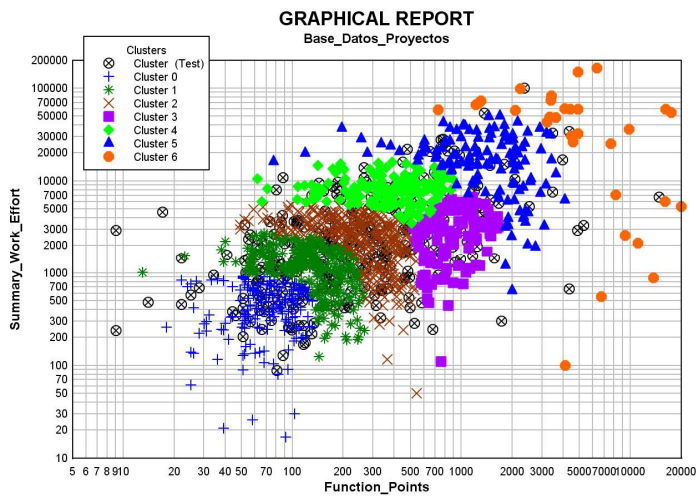


Fig. 1. Clusters given with RCT

Table 2. Statistics of the clusters obtained with RCT

	Cluster 0	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
No. Proj	177	306	321	114	169	128	31
Prob	0.13	0.23	0.24	0.095	0.15	0.105	0.027
Avg e	474.532	1257.227	2533.056	2892.66	7334.681	17849.586	46689.82
Avg fp	75.4	135.429	269.177	846.109	406.073	1495.645	6159.518
Std Dev e	252.23	614.61	1329.88	1646.11	3337.86	11052.25	38191.98
Std Dev fp	29.35	57.96	119.35	368.01	198.57	801.41	5062.28
Co coeff e-fp	0.10	-0.29	-0.47	0.31	0.03	-0.19	-0.33

Table 3 *a* and *b* constants are the linear log-log regression parameters obtained with the RCT tool and the MMRE and PRED(0.3) values. The first column of the table shows the values associated to all 1246 training projects without clustering.

**Table 3.** Linear Regression by means of double logarithmic transformation: *Linear log-log Plot (least-absolute-deviation)*

	Training Projects	CI 0	CI 1	CI 2	CI 3	CI 4	CI 5	CI 6
<i>a</i>	38.94	88.23	7783.84	43563.99	245.83	6649.36	128378.68	5763486.49
<i>b</i>	0.73	0.37	-0.38	-0.52	0.35	0.02	-0.28	-0.60
MMRE	1.58	0.96	0.55	0.74	0.9	0.26	0.95	15.1
PRED(0.3)	22.44%	45.19%	48.36%	46.72%	37.71%	61.53%	37.5%	32.25%

Finally, Table 4 describes a comparative analysis between the two models. The first column shows the MMRE and PRED(0.3) average predictive values, and at the second one appear the MMRE and PRED(0.3) model predictive values.

**Table 4.** Segmented model vs not segmented, using Linear log-log regression

	Curves Average Prediction		Model Prediction	
	MMRE	PRED(0.3)	MMRE	PRED(0.3)
RCT Linear log-log (l.a.d.)	0.72	46.16%	1.02	23.33%
M. Not Segmented. Linear log-log Plot (l.a.d.)	1.58	22.44%	1.41	22.34%

To obtain the average MMRE and PRED(0.3) values, the cluster 6 was not considered because the small number of projects and their scattered distribution; this can distort the results obtained.

## 5 Analysis of Results

In figure 1, the cluster 6 projects are very scattered, the function points range of values is 1,000-20,000, that point out a difficult regression curve fitting. There are other clusters that presents this characteristic, but less scattered.

Following with cluster 6, and watching table 3, we realize that the MMRE values are very high. The least absolute deviation method, supplies the best fit to clusters with high level of dispersion, which give less weight to remote points.

Respect to the model prediction values shown in table 4, there is not much difference between the segmented and not segmented parametric model. It can should to than the assignment of a project to a segment is not obvious. When you get a single mathematical model, there is no doubt about the equation to use in order to estimate de effort. But when there are several equations, as it occurs in the segmented model, it must exists a criterion that permits allocate a test project into the appropriate cluster. This criterion is crucial, and depending on it the prediction values of the segmented model could be better or worse. The criterium used in this paper is the minimum distance from the test instance with respect to the average of each cluster in the space, but not using the Euclidean metric but the Mahalanobis distance.

At the “Curves Average Prediction” column, the values corresponding to the segmented model improve in 200% the ones gotten by the not segmented model. The explanation of this is simple, as we have said previously, to analyze a homogeneous set of data offers better results than if the data are heterogeneous.



Consequently, the MMRE and PRED(0.3) values for the segmented model, which get a set of homogeneous clusters projects, are better than the gotten by the not segmented model, made up for heterogeneous projects.

## 6 Conclusions and Future Research Directions

With the inception of several organizations such as the ISBSG, there are a number of repositories of project management data. The problem faced by project managers is the large disparity of the their instances so that estimates using classical techniques are not accurate. The use of clustering techniques using data mining can be used to group instances from software engineering databases. In our case, this was used to provide segmented models such that each cluster had an associated estimation mathematical model. This has proven to be more accurate.

The comparison of using segmented parametric models against the unsegmented model has resulted in evidence that the parametric approach improve the curves average prediction values of MMRE and PRED(0.3). In general, it is better to use the set of segmented model curves provided by the cluster process, than a single model for all the database projects.

In segmented models, when we deal with a new project with unknown effort, the critical issue is to know the mathematical equation to use in order to get the effort estimation. If this question would be solved in a efficient manner, the model could be perfect, but this is not the case, and it is necessary to research about this aspect.

Further work will consist of, besides of research about projects assignment to clusters in order to get effort estimation, in using recursive clustering to improve, for example, the cluster 6 behavior, which have several scattered points. Recursive clustering can be applied to improve the clusters obtained at one step of the clustering process, getting other set of clusters of a lower level.

## References

1. Cuadrado, J.J., Sicilia, M.A., Garre, M., Rodríguez, D.: An empirical study of process-related attributes in segmented software cost-estimation relationships. *Journal of Systems and Software* 79(3), 351–361 (2006)
2. Garre, M., Cuadrado, J.J., Sicilia, M.A.: Recursive segmentation of software projects for the estimation of development effort. In: *Proceedings of the ADIS 2004 Workshop on Decision Support in Software Engineering*, CEUR Workshop proceedings, vol. 120 (2004)
3. Garre, M., Cuadrado, J.J., Sicilia, M.A., Charro, M., Rodríguez, D.: Segmented Parametric Software Estimation Models: Using the EM algorithm with the ISBSG 8 database. *Information Technology Interfaces*, Croacia ( junio 20-23, 2005)
4. Garre, M., Sicilia, M.A., Cuadrado, J.J., Charro, M.: Regression analysis of segmented parametric software cost estimation models using recursive clustering tool. In: Corchado, E.S., Yin, H., Botti, V., Fyfe, C. (eds.) *IDEAL 2006*. LNCS, vol. 4224, pp. 302–9743. Springer, Heidelberg (2006)
5. McLachlan, G., Krishnan, T.: *The EM Algorithm and Extensions*. Wiley series in probability and statistics. John Wiley & Sons (1997)

6. McLachlan, G., Peel, D.: *Finite Mixture Model*. Wiley, New York (2000)
7. Boehm, B., Abts, C., Sunita Chulani.: *Software Development Cost Estimation approaches – a survey*. USC Center for Software *Engineering Technical Report # USC-CSE-2000-505*
8. Dempster, A., Laird, N., Rubin, D.: Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society, Series B* 39(1), 1–38 (1977)
9. Oligny, S., Bourque, P., Abran, A., Fournier, B.: Exploring the relation between effort and duration in software engineering projects. In: *Proceedings of the World Computer Congress*, pp. 175–178 (2000)
10. Parametric Estimating Initiative. *Parametric Estimating Handbook*, 2nd ed. (1999)
11. NESMA, NESMA FPA Counting Practices Manual (CPM 2.0) (1996)
12. Dreger, J.B.: *Function Point Analysis*. Prentice Hall, Englewood Cliffs (1989)
13. Boehm, B.: *Software Engineering Economics*, vol. 10, Prentice-Hall (1981)
14. De Marco, T.: *Controlling Software Projects*. Yourdan Press (1982)
15. Conte, S.D., Dunsmore, H.E., Shen, V.Y.: *Software Engineering Metrics and Models*. Benjamin/Cumming Co., Inc., Menlo Park (1986)
16. Fenton, N.E.: *Software metrics: a rigorous approach*. Chapman & Hall, Londres (1991)
17. Fairley, R.E.: Recent advances in software estimation techniques. In: *International Conference on Software Engineering*, ACM, New York (1992)
18. Walkerdien, F., Jeffery, D.: Software cost estimation: A review of models, process, and practice. *Advances in Computers* 44, 59–125 (1997)
19. Wiczorek, I., Briand, L.: Resource estimation in software engineering, Technical Report, International Software Engineering Research Network (2001)
20. Idri, A., Abran, A.: Fuzzy Case-Based Reasoning Models for Software Cost Estimation (2002)
21. Idri, A., Abran, A., Khoshgoftaar, T.M.: Fuzzy Analogy: A new Approach for Software Cost Estimation. In: *Proceedings of the 11th International Workshop on Software Measurements*, Montreal, Canada, pp. 93–101 (2001)
22. Shepperd, M., Schofield, C.: Estimating software project effort using analogies. *IEEE Transactions on Software Engineering* (1997)
23. Dolado, J.J.: On the problem of the software cost function. *Information and Software Technology* 43, 61–72 (2001)
24. Dolado, J.J., Fernández, L.: Genetic Programming, Neural Networks and Linear Regression in Software Project Estimation, INSPIRE III, Process Improvement thorough Training and Education, pp. 155–171. The British Computer Society (1998)
25. Mair, C., Kadoda, G., Lefley, M., Keith, P., Schofield, C., Shepperd, M., Webster, S.: An investigation of machine learning based prediction systems. *The Journal of Systems and Software* 53, 23–29 (2000)
26. Briand, L., Langley, T., Wiczorek, I.: Using the European Space Agency Data Set: A Replicated Assessment and Comparison of Common Software Cost Modeling. In: *Proceedings of the 22th International Conference on Software Engineering*, Limerick, Ireland, pp. 377–386 (2000)
27. Briand, L.C., El Emam, K., Maxwell, K., Surmann, D., Wiczorek, I.: An Assessment and Comparison of Common Cost Software Project Estimation Methods. In: *Proc. International Conference on Software Engineering, ICSE 1999*, pp. 313–322 (1999)
28. Lee, A., Cheng, C.H., Balakrishnan, J.: Software development cost estimation: Integrating neural network with cluster analysis. *Information & Management* 34, 1–9 (1998)

# Author Index

Abran, Alain 168, 183  
Abu Talib, Manar 183  
Aquino, Nathalie 32

Barker, Mike 46  
Berry, Michael 1  
Buglione, Luigi 72

Condori-Fernández, Nelly 32  
Cruz-Lemus, José A. 129  
Cuadrado-Gallego, Juan J. 195

Daneva, Maya 60, 168  
Demirors, Onur 102  
Díaz-Ley, María 154  
Dumke, Reiner R. 95, 114

Farooq, Ayaz 114

García, Félix 154  
Garre, Miguel 195  
Gencel, Cigdem 72  
Genero, Marcela 129  
Gessenharter, Dominik 86  
Grau, Gemma 139

Idri, Ali 21  
Inoue, Katsuro 46

Johnson, Chris S. 1

Kassab, Mohamad 168  
Khelifi, Adel 183  
Kunz, Martin 95

Matsumoto, Ken-Ichi 46  
Matsumura, Tomoko 46  
Mendes, Emilia 21  
Merten, Alexander-Marc 86  
Mitani, Yoshiki 46

Ormandjieva, Olga 168, 183

Panach, Jose Ignacio 32  
Pastor, Oscar 32  
Piattini, Mario 129, 154  
Porta, Nicolas Fernando 86

Raschke, Alexander 86  
Rejas, Ricardo J. 195

Schmietendorf, Andreas 95  
Sicilia, Miguel-Ángel 195

Tarhan, Ayça 102  
Tsuruho, Seishiro 46

Valverde, Francisco 32

Zahi, Azeddine 21  
Zakrani, Abdelali 21