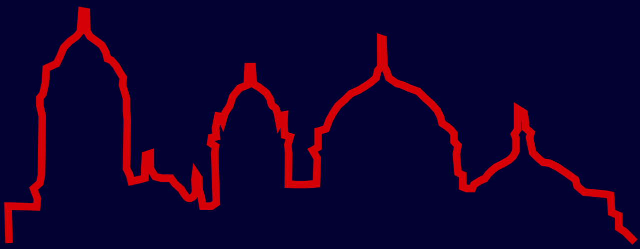
MENSURA 2006

PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON SOFTWARE PROCESS AND PRODUCT MEASUREMENT



November, 6-8, 2006
Cádiz, Spain

Alain Abran
Reiner Dumke
Mercedes Ruiz (Eds.)

PROCEEDINGS OF THE INTERNATIONAL CONFERENCE ON SOFTWARE PROCESS AND PRODUCT MEASUREMENT

MENSURA 2006

CÁDIZ – SPAIN

NOVEMBER, 6 – 8, 2006

UCA Universidad de Cádiz
Servicio de Publicaciones

Editors

Alain Abran
Department of Software and IT Engineering
École de technologie supérieure
1100, Notre-Dame Street West
Montréal, Québec
Canada H3C 1K3
E-mail: alain.abran@ele.etsmtl.ca

Reiner Dumke
Otto von Guericke University of Magdeburg
Postfach 4120
39016 Magdeburg
Germany
E-mail: dumke@ivs.cs.uni-magdeburg.de

Mercedes Ruiz
Research Group for Software Process Improvement and Formal Methods
Department of Computer Languages and Systems
University of Cádiz
Escuela Superior de Ingeniería
C/ Chile, 1
11003 – Cádiz
Spain
E-mail: mercedes.ruiz@uca.es

Edita: Servicio de Publicaciones de la Universidad de Cádiz
C/ Dr. Marañón, 3
11002 Cádiz
http://www.uca.es/publicaciones

ISBN13: 978-84-9828-101-9
ISBN10: 84-9828-101-6

Preface

This volume contents the papers presented at the Mensura 2006 conference held at Cádiz, Spain in November 2006 in conjunction with the 16th International Workshop on Software Measurement (IWSM2006) and MetriK Congress (MetriKon2006) hold at Postdam, Germany.

The objective of the International Conference on Software Process and Product Measurement (Mensura 2006) is to bring to light the most recent findings and results in the area of software measurement and to stimulate discussion between researchers and professionals.

The conference included top keynote speakers in the field of software measurement: Dr. Alain Abran from the University of Québec, Dr. Dieter Rombach from Fraunhofer Institute for Experimental Software Engineering and Mark Harman from King's College London.

We would like to thank the many people who have brought this International Conference into being: the Organizing Committee, the International Program Committee and the additional reviewers, particularly for all their hard work in reviewing both the abstracts and the final papers.

The organizers would also like to thank the University of Cádiz, the Research Group for Software Process Improvement and Formal Methods (SPI&FM) and the City Council of Cádiz for supporting the conference.

November 2006

Javier Dolado and Mercedes Ruiz
General and Program Chairs

Conference Organization

General Chair

José J. Dolado
University of the Basque Country, Spain

Program Chair

Mercedes Ruiz
University of Cádiz, Spain

Organizing Chair

Elena Orta
University of Cádiz, Spain

Program Committee

Sylvia Abrahao, Spain
Alain Abran, Canada
Javier Aroba, Spain
Moteoi Azuma, Japan
René Braungarten, Germany
Luigi Buglione, Italy
David Card, USA
Juan J. Cuadrado-Gallego, Spain
Jean-Marc Desharnais, Canada
José J. Dolado, Spain
Reiner R. Dumke, Germany
Christof Ebert, France
Nadine Hanebutte, USA
Hakim Lounis, Canada
Miguel Lopez, Belgium
John McGarry, USA
Roberto Meli, Italy
Pam Morris, Australia
John C. Munson, USA
Isabel Ramos, Spain
Daniel Rodriguez, UK
Tony Rollo, UK
Mercedes Ruiz, Spain
Grant Rule, UK
Andreas Schmientendorf, Germany

Stanimir Stojanov, Bulgaria
Charles Symons, UK
Javier Tuya, Spain
David Zubrow, USA
Horst Zuse, Germany

We would also like to thank the following persons who helped in reviewing the papers: Antònia Mas, Esperança Amengual, Elena Orta, Nuria Hurtado, Francisco Ruiz, Claudio de la Riva and Miguel Garre.

Table of Contents

Keynotes

Technical Papers

Authors Index

KEYNOTES

A Roadmap to Maturity for Software Measures

Alain Abran

École de Technologie Supérieure - ETS
1100 Notre-Dame Ouest, Montréal, Canada H3C 1K3
alain.abran@etsmtl.ca

Abstract. Up until recently 'software metrics' have been most often proposed as the quantitative tools of choice in software engineering, and the analysis of these had been most often discussed from the perspective referred to as 'measurement theory'. However, in other disciplines, it is the domain of knowledge referred to as 'metrology' that is the foundation for the development and use of measurement instruments. This paper presents an overview of the set of metrology concepts as documented in the ISO Vocabulary of Basic and General Terms in Metrology (VIM) and its use in analyzing 'software metrics'. It also presents the measurement coverage within the Guide to the Software Engineering Body of Knowledge (SWEBOK) as well as a proposed measurement body of knowledge. Throughout these analyses some gaps are identified which need to be addressed for software measurement to mature.

Keywords: Software Measures, Software Metrics, Metrology, SWEBOK, Measurement Body of Knowledge.

1 Introduction

In the field of software engineering, the term "metrics" is used in reference to multiple concepts, whether in terms of the quantity to be measured (measurand[1]), measurement procedures, measurement results or models of relationships across multiple measures, or of the objects themselves. In the software engineering literature, the term is applied, for instance, to a measure of a concept (e.g. McCabe cyclomatic complexity), to quality models (ISO 9126 – software product quality) and to estimation models (e.g. Halstead's effort equation, COCOMO estimation models). This has led to many curious problems, among them a proliferation of numerous publications on metrics for concepts of interest, but with a very low rate of acceptance and use by either researchers or practitioners, as well as a lack of consensus on how to validate so many proposals. The inventory of software metrics is at the present time so diversified and includes so many individual proposals that it is not seen to be economically feasible for either industry or the research community to investigate each of the hundreds of alternatives proposed to date.

[1] A measurand is defined as a particular quantity subject to measurement; the specification of a measurand may require statements about quantities such as time, temperature and pressure [ISO VIM].

In software engineering, the 'software metrics' approach has been up until fairly recently the dominant approach to software measurement. Most of these ‘metrics’ have been designed based either on the intuition of the researchers or on an empirical basis, or both. In their analysis of some of these ‘metrics’, researchers have most often used the concepts of 'measurement theory' as the foundation for their analytical investigation. However, while relevant, 'measurement theory' deals with only a subset of the classical set of concepts of measurement; 'software metrics' researchers, by focusing solely on measurement theory, have investigated mainly the representation conditions, the mathematical properties of the manipulation of numbers and the proper conditions for such manipulations [FENT97, ZUSE97].

Our survey of the literature on ‘software metrics’ has come up with almost no references to the classical concepts of metrology in these investigations into the quality of the ‘metrics’ proposed to the software engineering community. Only recently has some of the metrology related concepts been introduced in the ISO software engineering standards community. Is software measurement itself a mature tool set and can metrology help to investigate this research topic?

This paper presents in Section 2 the modeling of the sets of concepts in the ISO vocabulary of terms in metrology (VIM) and in section 3, examples of our use of the VIM in the analysis of ‘software metrics’. Section 4 presents an analysis of the coverage of measurement within SWEBOK and section 5, our proposed measurement body of knowledge to address some of the measurement gaps identified in the SWEBOK Guide. A discussion is presented in section 6.

2 Metrology

2.1 ISO Metrology concepts

In engineering as well as in other fields such as business administration and a significant number of the social sciences, measurement is one of a number of analytical tools. Measurement in these other sciences is based on a large body of knowledge; such a body of knowledge, built up over centuries and millennia, is commonly referred to as the field of 'metrology'. This domain is supported by government metrology agencies, which are to be found in most industrially advanced countries.

The ISO document that represents the official international and legal consensus is the ISO Vocabulary of Basic and General Terms used in Metrology [ISO VIM]. While this key ISO document is widely known in the field of metrology, it is almost unknown in the 'software metrics' community. This ISO VIM follows some of the concepts of the traditional presentation of vocabularies, with 120 terms described individually in textual descriptions. However, this mode of representation is challenging in terms of assembling the full set of interrelated terms; to improve the presentation and the understanding of this complex set of interrelated concepts, we presented in [ABRA02a] an initial set of models for the various levels of metrology concepts within the ISO Vocabulary.

The high-level model of the set of categories of terms is presented in Figure 1. This model, together with some sub-models presented later on, corresponds to our current understanding of the topology integrated into the vocabulary of this specialized area of the body of knowledge relating to metrology. To represent the relationships across the terms, the classical representation of a production process was selected: e.g. input, output and control variables, as well as the process itself inside the box. In Figure 1, the output is represented by the 'measurement results' and the process itself by the 'measurement' in the sense of measurement operations, while the control variables are the 'étalons'[2] (official yardsticks) and the 'quantities and units'. This set of concepts represents the 'measuring instrument'. It is to be noted that the measurement operations, and, of course, the measurement results, are influenced by the 'characteristics' of the measuring instruments. In the VIM, the term 'measurements' used as a single term corresponds to the 'set of operations' used for measuring.

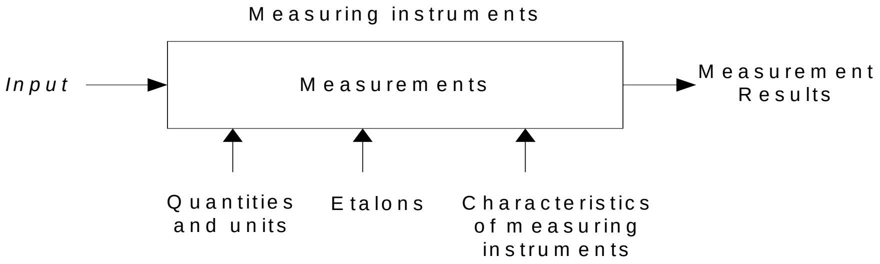


Figure 1: Model of the categories of metrology terms [ABRA02a]

The term 'metrology' includes all aspects of measurement (theoretical and practical), collectively referred to in the metrology literature as the science of measurement (Figure 2). Metrology encompasses the 'principles of measurement', which represent the scientific basis for measurement. From the principles of measurement, the 'method of measurement' in the general sense is then instantiated by a measurement as a set of operations. Figure 2 depicts this hierarchy of concepts.

The detailed topology of the measurement process is instantiated next in a 'measurement procedure' (Figure 3), again as a process model having the 'measurand' as its inputs, control variables and an output representing the 'measurement results'. To carry out a measurement exercise, an operator should design and follow a 'measurement procedure' which consists of a set of operations, specifically described, for the performance of a particular measurement according to a given measurement method. The instantiation of a measurement procedure handles a 'measurement signal' and produces a transformed value, which represents a given measurand. The results of the measurement can be influenced by an 'influence quantity' during the measurement

[2] Étalon: for instance, an internationally recognized material yardstick: the physical 'meter' etalon in length measurement recognized as the official 'étalon' for the meter. Étalons are also refined over time: for instance, the official definition of the meter has changed in 1983: it was then defined as the distance performed by the light, in an empty medium, in 1/299 792 458 second.

process: for example, the temperature of a micrometer during the measurement of the length of a particular object.

The category 'measurement results' is presented next in the form of a structured table according to the types of measurement results, the modes of verification of the measurement results and information about the uncertainty of measurement – Table 1.

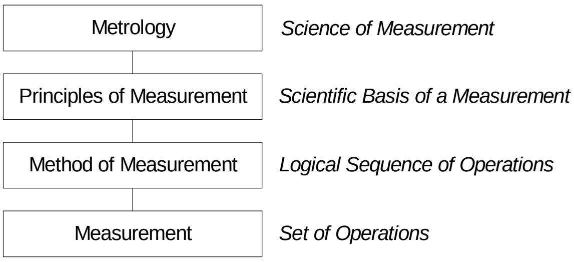

Metrology
Science of Measurement
Principles of Measurement
Scientific Basis of a Measurement
Method of Measurement
Logical Sequence of Operations
Measurement
Set of Operations

Figure 2: Measurement foundations [ABRA02a]

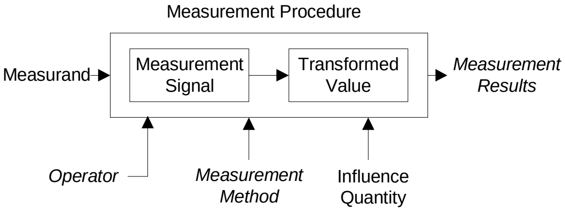


Figure 3: Measurement Procedure [ABRA02a]

Table 1: Classification of terms in the category of 'Measurement Results' [ABRA02a]

Types of measurement results	***Modes of verification of measurement results***	**Uncertainty of measurement**	
Indication (of a measuring instrument) Uncorrected result Corrected result	Accuracy of measurement Repeatability (of results of measurements) Reproducibility (of results of measurements)	Experimental standard deviation Error (of measurement) Deviation	Relative error Random error Systematic error Correction Correction factor

3 Metrology as a tool to analyze ‘software metrics’

3.1 Related work

While metrology has a long tradition of use in physics and chemistry, it is rarely referred to in the software engineering literature. Carnahan et al. [CARN97] are among the first authors to identify this gap in what they referred to as “IT metrology”; they highlight the challenges and opportunities arising from the application of the metrology concepts to information technology. In addition, they have proposed logical relationships between metrology concepts, consisting of four steps to follow to obtain measured values: defining quantity/attribute, identifying units and scales, determining the primary references and settling the secondary references. Moreover, Gray [GRAY99] discusses the applicability of metrology to information technology from the software measurement point of view.

[ABRA02b, 03] has highlighted some high-level ambiguities in the domain of software measurement, and proposed substituting the appropriate metrology terms for the current ambiguous and peculiar software metrics terminology unique to the domain of software engineering. In metrology, the term “metrics” is never used. Moreover, Sellami and Abran [SELA03] have investigated the contribution of metrology concepts to understanding and clarifying the framework for software measurement validation proposed by Kitchenham et al. in [KITC95].

3.2 Functional Size measurement

In the ISO software engineering community, there has been work to investigate and apply the metrology concepts to software measurement standards, including within specific measurement methods. The first type of measurement methods tackled at the ISO level were the functional size measurement methods with the publication of ISO meta-standards on functional size measurement [ISO 14143-1], dealing with some of the design issues of measurement method:

1. Part 1 of 14143: dealing with the ISO definitions of concepts for functional size measurement.
2. Part 2 of 14143: dealing with conformity assessment of the design of proposed functional size methods.
3. Part 3 of ISO 14143: dealing with the verification criteria of a functional size method to assist measurements users in selecting the methods most appropriate to their needs.
4. Part 4 of ISO 14143: providing a large set of functional user requirements against which candidate measurement methods can be tested.
5. Part 5 of ISO 14143: providing users with the information for analyzing which measurement method is most appropriate to the functional domain of the software to be measured.
6. Part 6 of ISO 14143: providing users with the information for selecting a specific measurement method according to their needs.

In addition, four specific methods have been recognized as ISO international standards, that is: IFPUG [ISO 20926], NESMA, MKII and COSMIC [ISO 19761], a second generation functional size measurement method. Many of the metrology related concepts have already been integrated into the design of the COSMIC method (ISO 19761), with particular attention paid to the characterization of the concept being measured, to the selected meta-model of the functionality, and to the units and quantities in the definition of the numerical assignment rules. Sellami and Abran [SELA05, ABRA97] have documented the metrology concepts addressed in ISO 19761 (COSMIC-FFP), both in the design of this measurement method and in some of its practical uses.

3.3 ISO 9126 'quality metrics'

The evaluation approach of ISO TR 9126-4 in [ABRA06] was based on an evaluation process previously used for investigating the design of cyclomatic complexity [ABRA04a], Halstead's metrics [ALQU05], IFPUG function points [ABR94, 96] and Use Case Points [OUW06].

The ISO 9126 series of documents on software product quality evaluation proposes a set of 120 metrics[3] for measuring the various characteristics and subcharacteristics of software quality. However, as is typical in the software engineering literature, the set of so-called metrics in ISO 9126 refers to multiple distinct concepts. To help in understanding and clarifying the nature of the metrics proposed in ISO TR 9126-4, each was analyzed from a metrology perspective and mapped to the relevant metrology concepts. Such an evaluation approach also contributes to identifying the measurement concepts that have not yet been tackled in the ISO 9126 series of documents. Each of these gaps represents an opportunity for improvement in the design and documentation of the measures proposed in ISO 9126. Based on the analysis using metrology concepts in [ABRA06] the following comments and suggestions were made:

- Identifying and classifying the 'quality in use metrics' into base and derived quantities makes it easy to determine which should be collected (base quantities) to be used subsequently in computing the other (derived) quantities.
- Some of the ISO 9126-4 derived units depend on other quantities with unknown units and are therefore ambiguous.
- None of the 'quality in use metrics' refers to any system of units, coherent (derived) unit, coherent system of units, international system of units (SI), off-system units, multiple of a unit, submultiple of a unit, true values, conventional true values or numerical values.
- None of the base and derived quantities, except for task time, has symbols for their measurement units.

[3] While the term "metrics" is used in ISO 9126, the use of this term will be abandoned and replaced by "measures" in the next ISO version currently in preparation as an initial step towards harmonizing the software engineering measurement terminology with the metrology terminology.

4 Measurement within SWEBOK

4.1 Measurement in SWEBOK

Measurement is embedded within the IEEE Computer Society definition of software engineering as

> “(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.
> (2) The study of approaches as in (1)” [IEEE 610].

The topic of measurement within SWEBOK was one of the editorial criteria for the initial write-up. The SWEBOK Knowledge Area editors were expected to adopt the position that the measurement 'theme' is common across all Knowledge Areas, and therefore had to be incorporated into the proposed breakdown of topics in each Knowledge Area. Since the acceptance criterion for inclusion in Guide to the SWEBOK was 'generally accepted', it is important to ask what did in fact gain an approval on a consensual basis with respect to measurement, and what can be learned from this consensus, or the lack of it. It is worth reminding that the 'generally accepted' definition adopted in SWEBOK originates from the Project Management Institute (PMI), that is: 'applies to most of the projects, most of the time, and widespread consensus validates its value and effectiveness'.

4.2 A measurement process model

In their work as ISO editors for the Guide to the Verification of Functional Size Measurement Methods [ISO 14143-3], Abran and Jacquet studied the various software engineering authors dealing with 'metrics validation' [JACQ97, 99]. Significant variations were found in the authors' approaches as well as the use of similar terms by these authors, but with very significant differences in the related concepts.

To clarify the confusion due to the inconsistent terminology used by these authors, a broader measurement process model was proposed (Figure 4) identifying 4 distinct steps, from the design of a measurement method to the exploitation of the measurement results [JACQ97, 99]. Then, the approaches of the various authors, as well as the validation concepts that were being addressed differently by these authors, were sorted out depending on whether or not they were addressing validation issues related to Steps 1 to 4 of the process model in Figure 4.

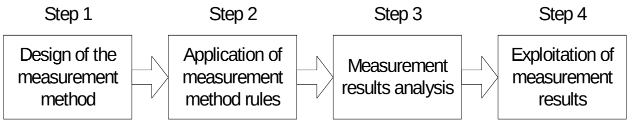

Step 1
Design of the measurement method
Step 2
Application of measurement method rules
Step 3
Measurement results analysis
Step 4
Exploitation of measurement results

Figure 4: Measurement Process – High-level Model

It is to be noted that very few of the measurement concepts present in the ISO VIM address the first step (design of a measurement method) and none address the last step (exploitation of the measurements results) of Figure 4. This is illustrated in Table 2, which depicts a partial mapping between Figures 1 and 4: for instance, for the design of a measurement method, the Abran and Jacquet model includes more concepts than simply 'quantities and units'.

Table 2: Alignment of metrology concepts with the measurement process model

Measurement process model [JACQ 97, 99]	**Step 1** Design of Measurement Methods	**Step 2** Application of measurement method rules	**Step 3** Measurement results analysis	**Step 4** Exploitation of measurement results
ISO metrology model [ABRA02]	• Quantities and units	• Measuring instruments • Characteristics of measuring instruments	• Measurement results	

4.3 Metrology concepts in SWEBOK

Using both the ISO set of metrology concepts model and the measurement process model [JACQ97, 99], the current status of the field of software measurements as documented in the SWEBOK Guide was analyzed [ABRA04b]. The results of the analysis of the presence of metrology concepts within each KA are presented in Table 3. Using a detailed inventory of the measurement-related statements appearing in the ten SWEBOK chapters, these statements were analyzed in terms of measurement concepts, and then mapped into both the set of metrology concepts presented in Section 2 and to the measurement process model presented in 4.2.

Table 3 lists, for each of the ten chapters of SWEBOK, which metrology concepts and measurement steps are addressed whenever a measurement-related statement appears in the SWEBOK Guide. From Table 3, it can be observed that a large majority of the measurement-related concepts mentioned in SWEBOK are listed in the category of concepts related to the exploitation of the measurement results. Very few SWEBOK statements directly address the measuring instrument or the quality of the direct measurement results (prior to their use in quantitative analytical models (assessment models or predictive models)). And only one measurement related statement in the Software Quality chapter addresses a single aspect of the design of measurement instrument, and only through a subset of the metrology concepts of quantities and units.

Table 3: Measurement steps and metrology category of concepts within SWEBOK [ABRA04b]

SWEBOK **Knowledge Area**	**Measurement** **Steps** **Topics**	**Step 1** Design of measurement methods (Quantities and units)	**Step 2** Application of measurement method rules (Measuring instruments)	**Step 3** Measurement results analysis	**Step 4** Exploitation of measurement results
Software engineering requirements	Process support and management				×
	Requirements negotiation				×
	Document quality				×
	Acceptance tests				×
	Requirements tracing				×
Software engineering design	Measures			×	
Software engineering testing	Evaluation of the program under test				×
	Evaluation of the tests performed				×
Software engineering maintenance	Software Maintenance Measurement				×
Software configuration management	Surveillance of software configuration management				×
Software engineering management	Goals				×
	Measurement Selection				×
	Measuring Software and its Development				×
	Collection of data		×		
	Software Measurement Models			×	
Software engineering process	Methodology in process measurement		×		
	Process Measurement Paradigms				×
Software engineering quality	Measuring the value of quality				×
	Fundamentals of Measurement	×			
	Measures			×	
	Measurement analysis techniques				×
	Defect characterization				×
	Additional Uses of SQA and V&V data				×

This highlights the fact that, even though the use of measurement results is quoted in most KA, both the KA editors and the extensive number of reviewers have not been able to come up and agreed on the availability of knowledge on measurement concepts which met the SWEBOK and PMI criteria of generally accepted, that is of 'applies to most of the projects, most of the time, and widespread consensus validates its value and effectiveness'. This does not mean that such other types of measurement knowledge do not exist in the literature, but rather that there is not yet a wide consensus on their value and effectiveness and their generalization power outside of the initial context of operations. It also points out to a significant lack software measurement methods with enough strengths as measurement instruments and meeting the metrology criteria for quality (of measuring instruments). Table 3 also points out to a lack of widely recognized and validated quantitative data to support yet the quality expected from an engineering viewpoint for the software engineering topics described.

This, of course, corresponds to a lack of recognized references to other measurement concepts from the recognized body of knowledge on metrology. This is a clear indication that, when looked at from an engineering perspective, measurement in software engineering is far from being mature and that it currently constitutes a fairly weak engineering foundation for the field of software engineering.

5 Measurement body of knowledge

Measurement is, of course, fundamental to the engineering disciplines, and, at the inception of the SWEBOK, it had been given to all the KA associate editors as a criterion for identifying relevant measurement-related knowledge in their respective Knowledge Areas. Individual associate editors initially developed each of the 10 Knowledge Areas on their own, and even though a large number of reviewers contributed in the numerous reviews, this still led to different levels of breadth and depth of treatment of subtopics like measurement: therefore, measurement-related knowledge has not been developed equally across Knowledge Areas. Subsequently, we proposed a unified view of the measurement knowledge in software engineering [ABRA04b, BUGL05] in the form of a proposal for a distinct KA on Software Measurement, taking into account all the measurement-related items from the 2004 version of the SWEBOK Guide [ABRA05]. This proposal is shown in Fig. 5.

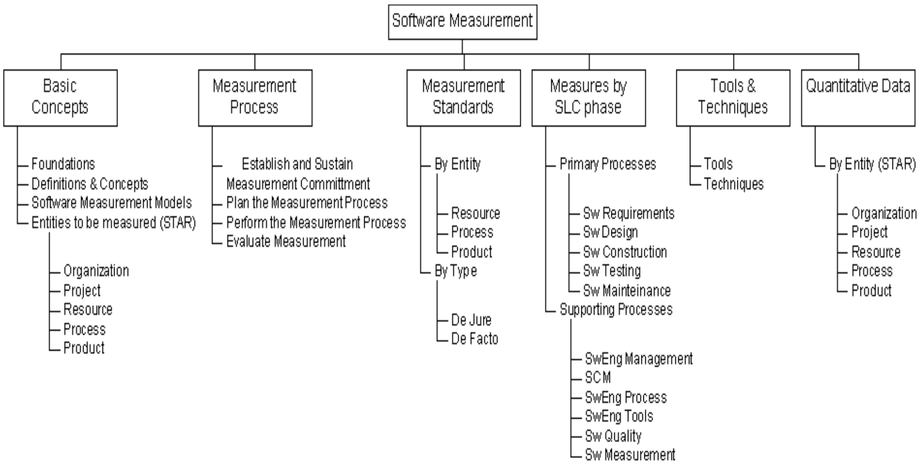


Figure 5 – Proposed breakdown for the Software Measurement KA [ABRA04b, BUGL05]

To investigate the credibility of the recommended reference material for our proposal for an additional KA on Software Measurement, the level of empirical support as documented in the SWEBOK references was investigated next, including twenty four (24) additional ones recommended in [BUGL05] to cover the “gaps” in the measurement references. These references have been grouped in three types:

- **International standards** (ISO, IEEE or other standards organizations): These are based on international consensus by either technical experts or ISO-recognized voting countries, or both.
- **Books**: These often represent only the author’s opinions. A book also contains a number of chapters, each of which could be based on a different type or types of empirical support.
- **Papers and book chapters**[4]: The most relevant empirical support method is mentioned. When there is not a direct mapping to one of the 12 empirical support methods proposed by [ZELK98], the “not applicable” code has been assigned.

To reach a point where this measurement body of knowledge would be recognized as generally accepted in the broader software engineering community, it is mentioned in [BUGL05] that further steps are required to get this measurement taxonomy validated by peers in the software engineering measurement community.

[4] This is an interim classification.

6 Discussion

Currently 'software metrics' are most often proposed as the measurement tools of choice in empirical studies in software engineering. While this field of 'software metrics' has most often been discussed from the perspective referred to as 'measurement theory', in other disciplines, however, it is the domain of knowledge referred to as 'metrology' that is the foundation for the development and use of measurement instruments and measurement processes. Measurement is recognized as a key element of engineering and, because of design criteria in the Guide to SWEBOK, it is pervasive in the Guide. But, is software measurement already a mature tool set?

In this paper, we have identified analytical tools to investigate the state of the art of measurement in software engineering, focusing on the set of metrology concepts. Both our initial modeling of the sets of measurement concepts documented in the ISO International Vocabulary of Basic and General Terms in Metrology and our measurement process model were used to survey, and position, the measurement-related statements in the Guide to the Software Engineering Body of Knowledge. This has revealed that, even though measurement-related statements appear throughout the SWEBOK document, they overwhelmingly concern the use of measurement results in assessment and predictive models. By contrast, there is in this document very little widely recognized validated knowledge from an engineering perspective, little on the quality of the quantitative inputs to these models, and almost nothing on supporting measuring instruments necessary to obtain these inputs.

Similarly, in the software engineering literature, even though there is a large number of 'metrics' proposed, there is still very little discussion on the topic of measuring instruments so overwhelmingly present in the traditional engineering disciplines.

This also illustrates that most of the metrology concepts, and sub-concepts, have not yet been discussed or addressed to a significant extent in the 'software metrics' literature. In the context where measuring instruments are necessary key elements of empirical studies, this points to a potentially significant weakness in current empirical studies in software engineering, while at the same time providing an indication of where metrology-related improvements in software measurement could contribute significantly to strengthening future empirical studies in software engineering.

This analysis from the metrology perspective suggests that the field of software measurement has not yet been fully addressed by current research, and that much work remains to be done to support software engineering as an engineering discipline based on quantitative data and adequate measurement methods meeting the classic set of criteria for measuring instruments as described by the metrology body of knowledge in large use in the engineering disciplines.

References

[ABRA94] Abran, Alain; Robillard, P.N., “Function Points: A Study of their Measurement Processes and Scale transformation,” Journal of Systems and Software, Vol. 65, 1994, pp. 171-184.

[ABRA96] Abran, Alain; Robillard, P.N., “Function Points Analysis: An Empirical Study of Its Measurement Processes,” IEEE Transactions on Software Engineering, Vol. 22, no. 12, December 1996, pp. 895-910.

[ABRA97] A. Abran and J.P. Jacquet, "A Structured Analysis of the New ISO Standard on Functional Size Measurement-Definition of Concepts", Fourth IEEE International Symposium and Forum on Software Engineering Standards, 1999, pp. 230-241.

[ABRA98] Abran, A., "Software Metrics Need to Mature into Software Metrology (Recommendations)," NIST Workshop on Advancing Measurements and Testing for Information Technology (IT), Maryland, USA, 1998.

[ABRA02a] A. Abran and A. Sellami, "Initial Modeling of the Measurement Concepts in the ISO Vocabulary of Terms in Metrology", in Software Measurement and Estimation, 12th International Workshop on Software Measurement - IWSM, Magdeburg (Germany) Oct. 7-9 2002, Shaker-Verlag, Aachen, 2002, ISBN 3-8322-0765-1, pp. 315.

[ABRA02b] A. Abran and A. Sellami, "Measurement and Metrology Requirements for Empirical Studies in Software Engineering", IEEE Software Technology and Engineering Practice Workshop (STEP), Montreal (Canada), 2002.

[ABRA03] Abran, A., Sellami, A., and Suryn, W., "Metrology, Measurement and Metrics in Software Engineering," 9th International Software Metrics Symposium, Sydney, Australia, 2003.

[ABR04a] Abran, A., Lopez, M., Habra, N., “An Analysis of the McCabe Cyclomatic Complexity Number,” 14th International Symposium on Software Measurement IWSM-Metrikon, Shaker Verlag, 2004, Magdeburg, Germany, pp. 391-405.

[ABRA04b] Abran, A.,Buglione, L, Sellami, A. (2004), “Software Measurement Body of Knowledge – Initial Validation using Vincenti’s Classification of Engineering Knowledge types”,14th International Workshop on Software Measurement IWSM-MetriKon 2004, Konigs Wusterhausen, Germany, Éditeur : Shaker-Verlag, pp. 255-270.

[ABRA05] A. Abran, J. Moore, P. Bourque, R. Dupuis, and L. Tripp, *Guide to the Software Engineering Body of Knowledge - SWEBOK - Version 2004*, IEEE-Computer Society Press: www.swebok.org, Los Alamitos (USA), 2005.

[ABRA06] Abran, A., Al Qutaish, R., Cuadrado, J., “Analysis of the ISO 9126 on Software Product Quality Evaluation form the Metrology and ISO 19539 Perspectives”, WSEAS Transactions on Computers, November Issue, Vol. 5, No. 11, 2006, World Scientific & Engineering Academy and Society (WSEAS), Greece, 2006, pp. 2778-2786. (ISSN: 1109-2750).

[ALQU05] Al Qutaish, R. E., Abran, A., “An Analysis of the Design and Definitions of Halstead’s Metrics,” 15th International Symposium on Software Measurement (IWSM), Shaker Verlag, 2005, Montréal, Canada.

[BUGL05] Buglione, L., Abran, A., “ Software Measurement Body of Knowledge – Overview of Empirical Support”, 15th International Workshop on Software Measurement – IWSM 2005, Montréal (Canada), Shaker Verlag, Sept. 12-14, 2005, pp. 353-368.

[CARN97] Carnahan, L., Carver, G., Gray, M., Hogan, M., Hopp, T., Horlick, J., Lyon, G., and Messina, E., "Metrology for Information Technology," *StandardView*, Vol. 5, No. 3, 1997, pp. 103-109.

[FENT97] N.E. Fenton and S.L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, second edition, PWS Publishing Co., 1997, pp. 638.

[GRAY99] Gray, M. M., "Applicability of Metrology to Information Technology," *Journal of Research of the National Institute of Standards and Technology*, vol. 104, No. 6, 1999, pp. 567-578.

[IEEE 610] IEEE IEEE 610.12: 1990, *IEEE Standard Glossary for Software Engineering Terminology*, IEEE Computer Society, USA, 1990.

[ISO VIM] ISO, *International Vocabulary of Basic and General Terms in Metrology*, International Organization for Standardization - ISO, Geneva, 1993, pp. 49.

[ISO 9126-4] ISO/IEC, *ISO/IEC TR 9126-4: Software Engineering - Product Quality - Part 4: Quality in Use Metrics*, International Organization for Standardization, Geneva, Switzerland, 2004.

[ISO 14143-1] ISO ISO/IEC 14143-1: 1998, *Information technology -- Software measurement -- Functional size measurement -- Part 1: Definition of concepts*, International Organization for Standardization - ISO, Geneva, 1998.

[ISO 14143-3] ISO ISO/IEC 14143-3: 2003, *Software Engineering- Functional Size Measurement- Part 3: Verification of Functional Size Measurement Methods*, International Organization for Standardization - ISO, Geneva, 2003.

[ISO 15939] ISO ISO/IEC 15939: 2002, *Information Technology - Software Engineering - Software Measurement Process*, International Organization for Standardization - ISO, Geneva, 2002.

[ISO 19761] ISO ISO/IEC 19761: 2003, *Software Engineering -- COSMIC-FFP -- A Functional Size Measurement Method*, International Organization for Standardization - ISO, Geneva, 2003.

[ISO 20926] ISO/IEC 20926: 2003, Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting Practices Manual, International Organization for Standardisation – ISO, Geneva, 2003.

[JACQ97] Jacquet, JP., Abran, A., "From Software Metrics to Software Measurement Methods: A Process Model", *International Software Engineering Standards Symposium and Forum, ISESS 97*: IEEE-Computer Society Press, 1997, pp. 128-135.

[JACQ99] Jacquet, JP., Abran, A., "Metric Validation Proposals : A Structured Analysis", in *Software Measurement - Current Trends in Research and Practice, 8th International Workshop on Software Measurement - IWSM 98*: Deutscher Universität Verlag, 1999, pp. 43-59.

[KITC95] Kitchenham, B., Pfleeger, S. L., and Fenton, N., "Towards a Framework for Software Measurement Validation," *IEEE Transaction on Software Engineering*, Vol. 21, No. 12, 1995, pp. 929-944.

[OUWE06] Ouwerkerk, J., Abran, A., ‘Evaluation of the Design of Use Case Points (UCP)’, MENSURA2006, Nov. 4-5, Shaker Verlag, 2006, Cadiz (Spain).

[SELA03] Sellami, A. and Abran, A., "The contribution of metrology concepts to understanding and clarifying a proposed framework for software measurement validation", 13th International Workshop on Software Measurement (IWSM), Montreal, Canada, Shaker Verlag, 2003, pp. 18-40.

[SELA05] Sellami, A., Abran, A., « Analyse d’une mesure de la taille fonctionnelle du logiciel selon le point de vue de la métrologie et application à la méthode COSMIC-FFP », Revue Génie Logiciel, Nanterre (France), No. 74, Sept. 2005, pp. 2-12.

[ZELK98] ZELKOWITZ M.V. & WALLACE D.R., *Experimental Models for validating Technology*, IEEE Computer, May 1998, pp. 23-31, URL: http://www2.umassd.edu/SWPI/science/r5023.pdf

[ZUSE97] Zuse H., *A Framework for Software Measurement*, Walter de Gruyter, Germany,Berlin, 1997, pp. 755

The Importance of Metrics in Search Based Software Engineering

Mark Harman

King's College London, Strand, London, WC2R 2LS

Abstract. This paper was written to accompany the author's keynote talk at the Mensura Conference 2006. The keynote will present an overview of Search Based Software Engineering (SBSE) and explain how metrics play a crucial role in SBSE.

1 Search Based Software Engineering (SBSE)

The aim of SBSE research is to move software engineering problems from human based search to machine-based search [15, 27]. SBSE uses a variety of techniques from the metaheuristic search, operations research and evolutionary computation paradigms. As a result, human effort moves up the abstraction chain, focusing on defining the fitness function that guides the automated search, rather than performing such a search manually. This keynote will argue that a fitness function is nothing more than a metric [24].

SBSE has been applied to a number of software engineering activities, right across the life-cycle from requirements engineering [4], project planning and cost estimation [1, 2, 3, 13, 19, 38] through testing [5, 6, 8, 11, 12, 23, 26, 39, 42, 55], to automated maintenance [9, 21, 25, 45, 46, 47, 50, 51], service-oriented software engineering [14], compiler optimization [16, 17] and quality assessment [10, 35]. The application of SBSE to software testing has recently witnessed intense activity to the point where, in 2004, there was sufficient material to warrant a survey paper on this sub-area of activity [43]. However, as the list of application areas above indicates, SBSE can be applied right across the spectrum of software engineering activity.

A wide range of different search techniques can and have been used in SBSE. The most widely used are local search (for example [38, 40, 45]), simulated annealing (for example [10, 28, 45]), genetic algorithms (for example [6, 22, 25, 55]) and genetic programming (for example [7, 18, 19, 20, 54]). However, no matter what search technique is employed, it is the fitness function that captures the crucial information; it determines a good solution from a poor one, thereby guiding the search. Several studies have concerned the analysis of widely used fitness functions and the fitness landscapes they denote [5, 29, 33, 36, 37, 38, 44, 49].

The term SBSE was coined in 2001 [27] and has since been the subject of several conferences and special issues. However, there was work on the application of search techniques in software engineering before this date [19, 34, 41, 48, 53, 56]. The first paper to apply search to software engineering applications that the author is aware of is by Xanthakis et al. [57].

2 Metrics in SBSE

Harman and Clark [24] identify four important properties in order for the SBSE approach to be successful. One is fairly obvious: there should be an absence of known optimal solutions; there is little point in searching for solutions if it is not possible to improve on the current best. Of course, for many software engineering problems, optimal solutions remain to be discovered.

Two of the other four 'Harman-Clark' properties relate to the search space: it should be sufficiently large to be interesting and worthy of automated search and it should have an approximate continuity so that search has effective guidance. However, the choice of fitness function also determines the shape of the fitness landscape; its peaks and troughs, its size and it continuity. Therefore, the definition of the fitness function plays a central role in SBSE research and practice.

The remaining requirement also concerns the fitness function used to guide the search. The fitness function is the metric that captures the observation that one element of the search space is better than another. All that is required is for the metric to order the elements of the search space. Therefore, in terms of measurement theory [52], the metric should measure an attribute of the candidate solutions on an ordinal scale. Fortunately this is one of the more primitive scales of measurement and almost all metrics in use in software engineering do measure on a scale that is at least as sophisticated as an ordinal scale.

This keynote will consider some metrics used in SBSE and will consider other metrics that may be useful. Hopefully this will act as a starting point for a discussion on the metrics that could be applied to future work on SBSE.

3 Acknowledgements

This keynote arose as a result of recent work undertaken by the author with many other colleagues in the growing Search Based Software Engineering community. The author's work is currently funded by a large EPSRC project, SEBASE (2006- 2011), for which the other partners are John Clark (University of York) and Xin Yao (University of Birmingham) and industrialists from DaimlerChrysler Berlin, Motorola UK and IBM UK. The author's work on Evolutionary testing is also supported by the EU Specific Targeted Research Project: EvoTest (2006- 2009). This keynote draws on ideas from these projects and from other keynotes and tutorials prepared by the author in collaboration with Joachim Wegener at DaimlerChrysler [30, 31, 32].

References

1. Jesús Aguilar-Ruiz, Isabel Ramos, José C. Riquelme, and Miguel Toro. An evolutionary approach to estimating software development projects. *Information and Software Technology*, 43(14):875–882, December 2001.
2. Giulio Antoniol, Massimiliano Di Penta, and Mark Harman. A robust search– based approach to project management in the presence of abandonment, rework, error and uncertainty. In 10th *International Software Metrics Symposium (METRICS 2004)*, pages 172–183, Los Alamitos, California, USA, September 2004. IEEE Computer Society Press.
3. Giulio Antoniol, Massimiliano Di Penta, and Mark Harman. Search-based techniques applied to optimization of project planning for a massive maintenance project. In 21st *IEEE International Conference on Software Maintenance*, pages 240–249, Los Alamitos, California, USA, 2005. IEEE Computer Society Press.
4. A.J. Bagnall, V.J. Rayward-Smith, and I.M. Whittley. The next release problem. *Information and Software Technology*, 43(14):883–890, December 2001.
5. André Baresel, David Wendell Binkley, Mark Harman, and Bogdan Korel. Evolutionary testing in the presence of loop–assigned flags: A testability transformation approach. In *International Symposium on Software Testing and Analysis (ISSTA 2004)*, pages 108–118, Omni Parker House Hotel, Boston, Massachusetts, July 2004. Appears in Software Engineering Notes, Volume 29, Number 4.
6. André Baresel, Harmen Sthamer, and Michael Schmidt. Fitness function design to improve evolutionary structural testing. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1329–1336, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.
7. Terry Van Belle and David H. Ackley. Code factoring and the evolution of evolvability. In GECCO 2002: *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1383–1390, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.
8. Leonardo Bottaci. Instrumenting programs with flag variables for test data search by genetic algorithms. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1337–1342, New York, 9-13 July 2002. Morgan Kaufmann Publishers.
9. Salah Bouktif, Giuliano Antoniol, Ettore Merlo, and Markus Neteler. A novel approach to optimize clone refactoring activity. In *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 2, pages 1885–1892, Seattle, Washington, USA, 8-12 July 2006. ACM Press.
10. Salah Bouktif, Houari Sahraoui, and Giuliano Antoniol. Simulated annealing for improving software quality prediction. In *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 2, pages 1893– 1900, Seattle, Washington, USA, 8-12 July 2006. ACM Press.
11. Lionel C. Briand, Jie Feng, and Yvan Labiche. Using genetic algorithms and coupling measures to devise optimal integration test orders. In *SEKE*, pages 43–50, 2002.
12. Lionel C. Briand, Yvan Labiche, and Marwa Shousha. Stress testing real-time systems with genetic algorithms. In Hans-Georg Beyer and Una-May O'Reilly, editors, *Genetic and Evolutionary Computation Conference, GECCO 2005*, Proceedings, Washington DC, USA, June 25-29, 2005, pages 1021–1028. ACM, 2005.
13. Colin J. Burgess and Martin Lefley. Can genetic programming improve software effort estimation? A comparative evaluation. *Information and Software Technology*, 43(14):863–873, December 2001.
14. Gerardo Canfora, Massimiliano Di Penta, Raffaele Esposito, and Maria Luisa Villani. An approach for qoS-aware service composition based on genetic algorithms. In Hans-Georg Beyer and Una-May O'Reilly, editors, *Genetic and Evolutionary Computation Conference,*

GECCO 2005, Proceedings, Washington DC, USA, June 25-29, 2005, pages 1069–1075. ACM, 2005.

15. John Clark, José Javier Dolado, Mark Harman, Robert Mark Hierons, Bryan Jones, Mary Lumkin, Brian Mitchell, Spiros Mancoridis, Kearton Rees, Marc Roper, and Martin Shepperd. Reformulating software engineering as a search problem. *IEE Proceedings — Software*, 150(3):161–175, 2003.
16. Myra Cohen, Shiu Beng Kooi, and Witawas Srisa-an. Clustering the heap in multi-threaded applications for improved garbage collection. In *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 2, pages 1901–1908, Seattle, Washington, USA, 8-12 July 2006. ACM Press.
17. Keith D. Cooper, Philip J. Schielke, and Devika Subramanian. Optimizing for reduced code space using genetic algorithms. In *Proceedings of the ACM Sigplan 1999 Workshop on Languages, Compilers and Tools for Embedded Systems (LCTES'99)*, volume 34.7 of ACM Sigplan Notices, pages 1–9, NY, May 5 1999. ACM Press.
18. Jose J. Dolado. On the problem of the software cost function. *Information and Software Technology*, 43(1):61–72, 1 January 2001.
19. José Javier Dolado. A validation of the component-based method for software size estimation. *IEEE Transactions on Software Engineering*, 26(10):1006–1021, 2000.
20. Maria Cludia Figueiredo Pereira Emer and Silva Regina Vergilio. GPTesT: A testing tool based on genetic programming. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1343–1350, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.
21. Deji Fatiregun, Mark Harman, and Rob Hierons. Search-based amorphous slicing. In 12th *International Working Conference on Reverse Engineering (WCRE 05)*, pages 3–12, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA, November 2005.
22. Nicolas Gold, Mark Harman, Zheng Li, and Kiarash Mahdavi. A search based approach to overlapping concept boundaries. In 22nd *International Conference on Software Maintenance (ICSM 06)*, page To appear, Philadelphia, Pennsylvania, USA, September 2006.
23. Qiang Guo, Robert Mark Hierons, Mark Harman, and Karnig Derderian. Constructing multiple unique input/output sequences using evolutionary optimization techniques. *IEE Proceedings — Software*, 152(3):127–140, 2005.
24. Mark Harman and John Clark. Metrics are fitness functions too. In 10th *International Software Metrics Symposium (METRICS 2004)*, pages 58–69, Los Alamitos, California, USA, September 2004. IEEE Computer Society Press.
25. Mark Harman, Robert Hierons, and Mark Proctor. A new representation and crossover operator for search-based optimization of software modularization. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1351–1358, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.
26. Mark Harman, Lin Hu, Robert Mark Hierons, JoachimWegener, Harmen Sthamer, Andr Baresel, and Marc Roper. Testability transformation. *IEEE Transactions on Software Engineering*, 30(1):3–16, January 2004.
27. Mark Harman and Bryan F. Jones. Search based software engineering. *Information and Software Technology*, 43(14):833–839, December 2001.
28. Mark Harman, Kathleen Steinhöfel, and Alexandros Skaliotis. Search based approaches to component selection and prioritization for the next release problem. In 22nd *International Conference on Software Maintenance (ICSM 06)*, page To appear, Philadelphia, Pennsylvania, USA, September 2006.
29. Mark Harman, Stephen Swift, and Kiarash Mahdavi. An empirical study of the robustness of two module clustering fitness functions. In *Genetic and Evolutionary Computation Conference (GECCO 2005)*, Washington DC, USA, June 2005. Association for Computer Machinery. to appear.

30. Mark Harman and Joachim Wegener. Evolutionary testing. *In Genetic and Evolutionary Computation (GECCO)*, Chicago, July 2003.
31. Mark Harman and Joachim Wegener. Getting results with search–based software engineering. In *26th IEEE International Conference and Software Engineering (ICSE 2004)*, Los Alamitos, California, USA, 2004. IEEE Computer Society Press. To Appear.
32. Mark Harman and Joachim Wegener. Search based testing. In 6th *Metaheuristics International Conference (MIC 2005)*, Vienna, Austria, August 2005. To appear.
33. E. Hart and P. Ross. GAVEL - a new tool for genetic algorithm visualization. *IEEE-EC*, 5:335–348, August 2001.
34. B.F. Jones, H.-H. Sthamer, and D.E. Eyres. Automatic structural testing using genetic algorithms. *The Software Engineering Journal*, 11:299–306, 1996.
35. T. M. Khoshgoftaar, Liu Yi, and N. Seliya. A multiobjective module-order model for software quality enhancement. *IEEE Transactions on Evolutionary Computation*, 8(6):593–608, December 2004.
36. Yong-Hyuk Kim and Byung-Ro Moon. Visualization of the fitness landscape, A steady-state genetic search, and schema traces. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, page 686, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.
37. Yong-Hyuk Kim and Byung-Ro Moon. New usage of sammon's mapping for genetic visualization. In *Genetic and Evolutionary Computation – GECCO-2003*, volume 2723 of LNCS, pages 1136–1147, Berlin, 12-16 July 2003. Springer-Verlag.
38. Colin Kirsopp, Martin Shepperd, and John Hart. Search heuristics, case-based reasoning and software project effort prediction. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1367–1374, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.
39. Zheng Li, Mark Harman, and Rob Hierons. Meta-heuristic search algorithms for regression test case prioritization. *IEEE Transactions on Software Engineering*. To appear.
40. Kiarash Mahdavi, Mark Harman, and Robert Mark Hierons. A multiple hill climbing approach to software module clustering. In *IEEE International Conference on Software Maintenance*, pages 315–324, Los Alamitos, California, USA, September 2003. IEEE Computer Society Press.
41. Spiros Mancoridis, Brian S. Mitchell, C. Rorres, Yih-Farn Chen, and Emden R.Gansner. Using automatic clustering to produce high-level system organizations of source code. In *International Workshop on Program Comprehension (IWPC'98)*, pages 45–53, Los Alamitos, California, USA, 1998. IEEE Computer Society Press.
42. Phil McMinn, Mark Harman, David Binkley, and Paolo Tonella. The species per path approach to search-based test data generation. In *International Symposium on Software Testing and Analysis (ISSTA 06)*, pages 13–24, Portland, Maine, USA, 2006.
43. Philip McMinn. Search-based software test data generation: A survey. *Software Testing, Verification and Reliability*, 14(2):105–156, June 2004.
44. Brain S. Mitchell. *A Heuristic Search Approach to Solving the Software Clustering Problem*. PhD Thesis, Drexel University, Philadelphia, PA, January 2002.
45. Brian S. Mitchell and Spiros Mancoridis. Using heuristic search techniques to extract design abstractions from source code. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1375–1382, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.
46. Brian S. Mitchell and Sprios Mancoridis. On the automatic modularization of software systems using the bunch tool. *IEEE Transactions on Software Engineering*, 32(3):193–208, 2006.
47. Mark O'Keeffe and Mel OCinneide. Search-based software maintenance. In *Conference on Software Maintenance and Reengineering (CSMR'06)*, pages 249–260, March 2006.

48. R. P. Pargas, M. J. Harrold, and R. R. Peck. Test-data generation using genetic algorithms. *The Journal of Software Testing, Verification and Reliability*, 9:263– 282, 1999.

49. Hartmut Pohlheim. Visualization of evolutionary algorithms - set of standard techniques and multidimensional visualization. In Wolfgang Banzhaf, Jason Daida, Agoston E. Eiben, Max H. Garzon, Vasant Honavar, Mark Jakiela, and Robert E. Smith, editors, *Proceedings of the Genetic and Evolutionary Computation Conferenc*e, volume 1, pages 533–540, San Francisco, CA 94104, USA, 13-17 July 1999. Morgan Kaufmann.

50. Olaf Seng, Markus Bauer, Matthias Biehl, and Gert Pache. Search-based improvement of subsystem decompositions. In Hans-Georg Beyer and Una-May O'Reilly, editors, *Genetic and Evolutionary Computation Conference, GECCO 2005*, Proceedings, Washington DC, USA, June 25-29, 2005, pages 1045–1051. ACM, 2005.

51. Olaf Seng, Johannes Stammel, and David Burkhart. Search-based determination of refactorings for improving the class structure of object-oriented systems. *In GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 2, pages 1909–1916, Seattle, Washington, USA, 8-12 July 2006. ACM Press.

52. Martin J. Shepperd. *Foundations of software measurement*. Prentice Hall, 1995.

53. Nigel Tracey, John Clark, and Keith Mander. The way forward for unifying dynamic test-case generation: The optimisation-based approach. In *International Workshop on Dependable Computing and Its Applications (DCIA)*, pages 169–180. IFIP, January 1998.

54. StefanWappler and JoachimWegener. Evolutionary unit testing of object-oriented software using strongly-typed genetic programming. In *GECCO 2006: Proceedings of the 8th annual conference on Genetic and evolutionary computation*, volume 2, pages 1925–1932, Seattle, Washington, USA, 8-12 July 2006. ACM Press.

55. Joachim Wegener, André Baresel, and Harmen Sthamer. Evolutionary test environment for automatic structural testing. *Information and Software Technology Special Issue on Software Engineering using Metaheuristic Innovative Algorithms*, 43(14):841–854, 2001.

56. Joachim Wegener, K. Grimm, M. Grochtmann, Harmen Sthamer, and Bryan F. Jones. Systematic testing of real-time systems. In *4th International Conference on Software Testing Analysis and Review (EuroSTAR 96)*, 1996.

57. S. Xanthakis, C. Ellis, C. Skourlas, A. Le Gall, S. Katsikas, and K. Karapoulios. Application of genetic algorithms to software testing (Application des algorithms géenétiques au test des logiciels). In *5th International Conference on Software Engineering and its Applications*, pages 625–636, Toulouse, France, 1992.

TECHNICAL PAPERS

An Assessment of Function Point-Like Metrics for Object-Oriented Open-Source Software

Vieri Del Bianco[1], Luigi Lavazza[1,2]

[1] University of Insubria, Via Mazzini, 5
21100 Varese (Italy)
[2] CEFRIEL, Via Fucini, 2
20133 Milano (Italy)

delbian1970@yahoo.it, luigi.lavazza@uninsubria.it

Abstract. Since object-oriented programming became a popular development practice, researchers and practitioners have defined several techniques aimed at measuring object-oriented software. Among these, several Function Point-like approaches have been proposed. However, mapping the concepts at the basis of Function Point Analysis onto object-oriented concepts is not straightforward; therefore, there is the need to test the validity of FP-based object-oriented metrics. This article addresses questions concerning the relations between FP-like object-oriented metrics and more traditional object-oriented metrics, and the relations of object-oriented FP-like metrics with each other. The work reported here is based on the analysis of two sets of object-oriented applications: a set of programs developed by master students of a software engineering course and a set of open-source programs. Different kinds of FP-based object-oriented metrics were applied, and the results analyzed.

Keywords: Software measurement, object-oriented metrics, Function Points, empirical validation.

1 Introduction

Since the introduction of object-oriented programming in the industrial practice, several techniques aimed at measuring object-oriented software have been proposed. Metrics were proposed to evaluate the characteristics of object-oriented design [5], to evaluate the characteristics of UML [9], to estimate the development effort [2][6], and for several other purposes.

Among the proposed metrics, a number of approaches undertook the adaptation of the principles of Function Point Analysis [1] to object-oriented systems. However, little experimental results are available to prove the effectiveness of these metrics. In fact –as noted in [4]– new measures are continuously proposed, while existing ones have not yet been satisfactorily investigated.

This paper aims at developing experimental evidence concerning Function Point-like object-oriented metrics through the application of such metrics to a set of programs. In fact, previous work carried out by the authors [7] provided some initial evaluation of the expressiveness of two among the most relevant FP-like object-oriented metrics, namely Object-Oriented Function Points (OOFPs) [2,3] and Class Points [6]. In particular, it was found that:

- The two FP-like object-oriented metrics that were evaluated seem to be substantially equivalent, since they appear to be linearly related.
- Both FP-like object-oriented metrics that were evaluated are computed –according to their definition– by means of rather complex procedures. Nevertheless, they appear to be equivalent to the linear combination of basic object-oriented metrics like the number of attributes and methods.

The work done suggested that the validity of the FP-like approaches proposed for measuring the "functional" dimensions of object-oriented software needed further validation. In any case, since the analysis reported in [7] concerned only programs developed by students as part of a course, it was necessary to experimentally apply the proposed metrics also to "real life" object-oriented system.

In order to address such programs, we resorted to Open Source Software (OSS). We selected a set of open source programs implementing chat-like communication applications of the same type of those previously analyzed in [7].

This paper reports the results of the measurements, highlighting the relations existing among FP-like OO metrics, traditional OO metrics (e.g., number of classes, attributes, methods, etc.), and lines of code (LOCs).

The paper is organized as follows: section 2 describes the analyzed systems and the computation of different FP-like object-oriented metrics; section 3 is the core of the paper: it reports about the analysis of the measured data and gives evaluations, comparisons and discussions about the FP-like object-oriented metrics and the traditional object-oriented metrics. Section 4 draws some conclusions.

2 Empirical Evaluation

The empirical evaluation concerns two sets of programs. The first one includes the programs developed by master students of a software engineering course. Students were required to develop a simple –but not trivial– chatting system. The development was meant to test the ability of the students in object-oriented design and implementation (using UML and Java). The second set of programs includes open source programs (all licensed under the GNU Public License) implementing chatting systems or similar functionality. These programs were downloaded from the SourceForge repository (http://sourceforge.net/). They are:

Llamachat. This program provides an open source chat server/client pair for use on the web. It is written in Java and supports several chat functionality including secure connections, emoticons, administrative class users, and more. Additional information on Llamachat is available from http://sourceforge.net/projects/llamachat/

ChipChat. This is a web chat program written in Java. It is a web application and needs a web server such as Tomcat. It does not use the HTTP refresh feature, instead it uses applets for communication with the server and javascript for smooth screen updating. It requires no installation at the client. Additional information is available from http://chipchat.sourceforge.net/

Chat Everywhere. This program lets users establish a real-time discussion forum. It features extended (Irc-like) commands, several levels of hierarchy and easy configurability. Additional information on Chat Everywhere is available from http://chateverywhere.sourceforge.net/

FreeCS. This is a free chatserver written in Java. It features customizable layout and authentication by authorization. The non-blocking-IO-Classes of Java (java.nio) are used for networking. More on FreeCS at http://freecs.sourceforge.net/

Although in principle the considered FP-like metrics could be applied also to the results of the design stage (e.g., to UML diagrams), we employed them exclusively to measure code, since the design artifacts were not available.

Among the object-oriented FP-based approaches we considered OOFPs proposed by Antoniol et al. [2] and Class Points, proposed by Costagliola et al. [6]. These approaches were chosen because they represent two opposites in the wide range of object-oriented FP-based metrics. In facts, OOFPs try to adhere as closely as possible to the IFPUG definition of FPs; actually, OOFPs are proposed as a replacement for IFPUG FPs. On the contrary, class points –although inspired by FPs– do not strive to adhere to the classical FP approach, but just adopt the concepts underlying the FPA.

2.1 Computation of the OOFPs.

The computation of OOFPs follows the function point counting procedure. Classes within the application boundary correspond to ILFs, while classes outside the application boundary (including libraries) correspond to EIFs.

Inputs, Outputs and Inquiries are all treated in the same way: they are called generically “service requests” and correspond to class methods. Note that the term “service request” used here is proposed in [2]; nevertheless, it actually indicates method definitions. On the contrary, “service requests” in the definition of Class Points indicate service invocations.

The complexity of ILFs and EIFs depends on the number and type of attributes and associations. The complexity of service requests depends on the number and type of method parameters. Function types contribute to the FPs according to the weights defined by Albrecht [1]. For additional details refer to the paper by Antoniol et al. [2]. In the original definition of OOFPs, several ways of considering class aggregates and generalization hierarchies are proposed. Here we use the simplest among the proposed approaches, which counts every class as either an EIF or an ILF.

The data for the computation of OOFPs were derived automatically from Java code by means of the tool described in [8].

2.2 Computation of Class Points.

Class points were computed according to the indications provided by Costagliola et al. [6]. In brief, for every class the number of services required (NSR), the number of external methods (NEM) and the number of attributes (NOA) are computed. The complexity of a class is then evaluated on the basis of its NSR, NEM and NOA. Then, classes are weighted according to their complexity and type. The class types are the following: Problem Domain (PDT), Human Interaction (HIT), Data Management (DMT), Task Management (TMT). The sum of the data representing the weighed complexity of classes gives the number of class points.

There are two definitions of class points: one that does not take into account the number of attributes, and one that does and is thus expected to represent more precisely the size of the system. Since the number of attributes was available –having already been counted for the computation of OOFPs– we adopted the definition which includes the number of attributes.

In the original paper by Costagliola et al. [6] it was considered the possibility of adjusting the Class Point count in order to make effort prediction based on CPs more precise. Since we are not interested in using class points for effort estimation, we do not take into consideration any sort of adjustment. In the rest of the paper we use the term "class points" (or CP, or TUCP) to indicate the Total Unadjusted Class Points.

Table 1. The measures of the chatting systems.

Application	eLOC	NoC	NoM	NoA	OOFP	TUCP
chateverywhere_java	1747	24	142	104	877	122
chateverywhere_swing	1868	27	171	124	1026	133
Llamachat	1275	33	134	112	582	149
Chipchat	1381	14	114	70	578	76
Freecs	13699	128	978	749	3790	613
BCF2	627	15	82	35	370	62
CC2	796	22	102	32	482	71
FDT2	884	10	78	49	419	41
MDP2	1441	32	196	125	852	110
RER2	733	16	124	55	485	72
SM2	697	17	88	41	387	69
BCF1	406	13	57	23	266	51
CC1	631	21	80	33	454	69
FDT1	657	10	55	46	354	43
MDP1	1143	26	149	106	720	92
RER1	465	12	78	41	334	47
SM1	378	12	58	22	263	46

3 Results

The available chatting systems were measured, in order to characterize each of them with respect to: effective LOCs (i.e., excluding comment lines, blank lines and lines with only syntactic relevance), number of classes (NoC), methods (NoM), and attributes (NoA), OOFPs, and CPs. Table 1 reports the data obtained from the measurement of the available chatting systems. The top 5 lines refer to the open source applications, while the bottom lines refer to the students' applications. For each of these, two versions having different functionalities were released. So, for instance, BCF1 and BCF2 refer to the two versions of program BCF.

The data reported in Table 1 were derived directly from the Java code of the applications, by means of an automatic measurement tool [8].

3.1 Relations between OOFPs and traditional object-oriented metrics

The statistical analysis performed in [7] showed a strong linear correlation between basic metrics (i.e., number of attributes, methods, etc.) and OOFPs. This was considered particularly relevant, since it generated the doubt that an ambitious metric, also relatively complex to compute, such as the OOFPs was actually equivalent to much more simple and straightforward metrics, namely the number of methods.

In order to explore this issue for OSS, we employed the model derived from the analysis of students' programs to predict the OOFPs of open source programs. Following the indications provided by [7], we considered the dependence of OOFPs on the number of methods (NoM) alone.

From the available data concerning the students' programs we derived the following simple model:

OOFP = 4.6 * NoM

Such model was applied to predict the OOFPs of open source programs. The result was that OOFPs of OSS can be predicted quite well, with a 17% average absolute error (errors varying in the 6-26% range).

Fig. 1 illustrates the position of the measured values of OOFPs of open source chat programs with respect to the model derived from the students' programs (OOFP = 4.6 * NoM). It is quite easy to see that al the OS programs are very close to the model.

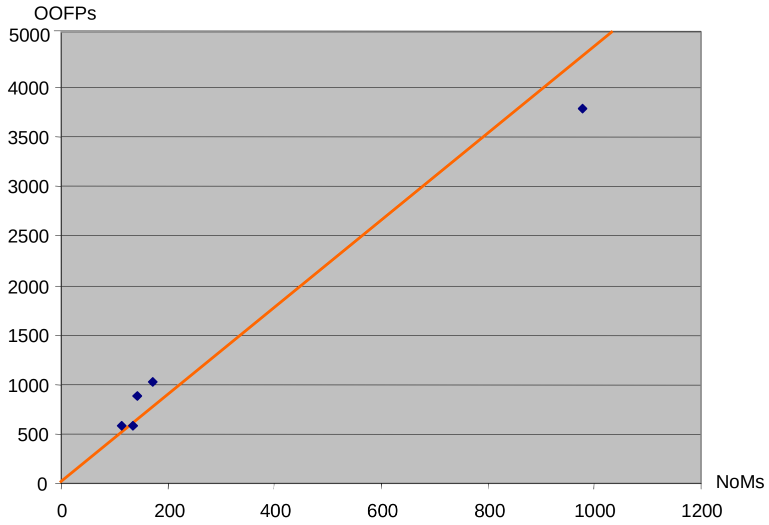


Fig. 1. Correlation of OOFPs with respect to the Number of Methods.

3.2 Relations between TUCPs and basic object-oriented metrics

The statistical analysis performed in [7] showed a strong linear correlation between basic metrics (namely the number of methods) and TUCPs as well. From the available data concerning the students' programs we derived the following simple model:

TUCP = 0.64 * NoM

Such model was applied to predict the TUCPs of open source programs. The result was that TUCPs of OSS can be predicted very well, with just a 3% average absolute error (errors varying in the 0.4-10.8% range).

Fig. 2 illustrates the position of the measured values of TUCPs of open source chat programs with respect to the model derived from the students' programs (TUCP = 0.64 * NoM). Also in this case it is quite easy to see that al the OS programs are very close to the model.

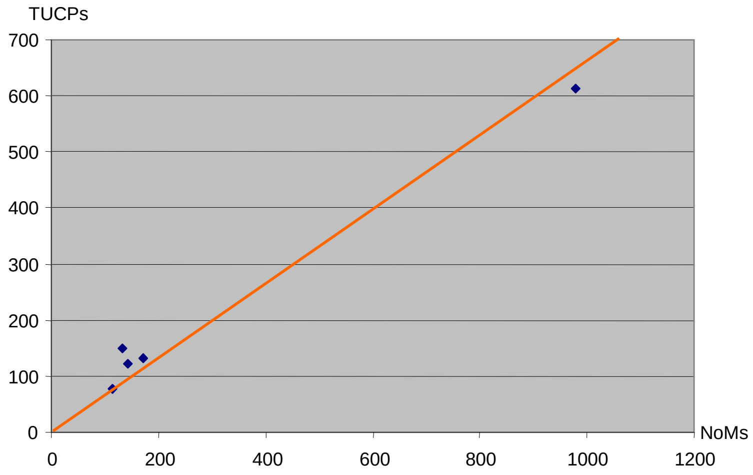


Fig. 2. Correlation of TUCPs with respect to the Number of Methods.

3.3 Relations between TUCPs and OOFPs

According to the results reported above, there is a clear dependence of both OOFPs and TUCPs on the number of methods. Therefore, we expect that a correlation between OOFPs and TUCPs exists too. We therefore analyzed the available data in order to find a possible relation between TUCPs and OOFPs.

A very strong linear correlation (r^2=0.98) was found. In fact it was quite easy to observe, even without the help of any sophisticated statistical tool, that OOFP are generally between 6 and 7 times the TUCPs.

There was only one noticeable exception. For the chat software system `llamachat` the OOFPs are less than 4 times the TUCPs. This difference is probably due to an unusual organization of the program. In fact, by inspecting the code, we found that this is a very well modularized program, with cleanly separated and factorized classes. The program also exploits an unusually large number libraries, but without hiding the dependency of the program from these libraries. As a consequence, the program features an unusually low number of ILF and an unusually high number of EIF. This ends up in a “strange” ratio between UUFPs and TUCPs.

Fig. 3 shows the correlation between TUCPs and OOFPs.

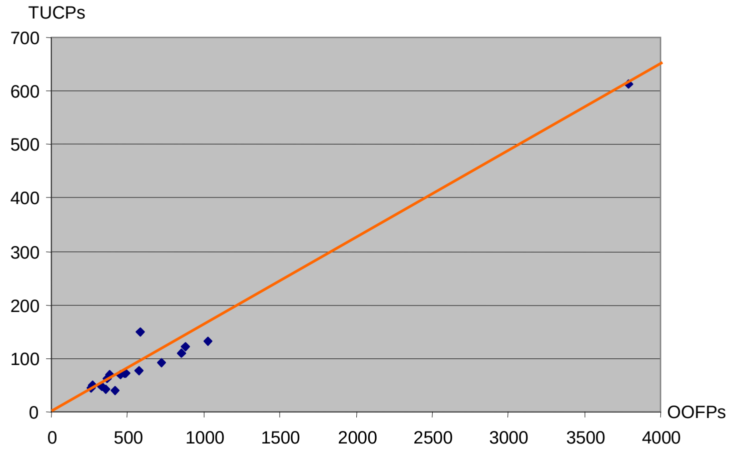


Fig. 3. Correlation of TUCPs with respect to OOFPs.

4 Conclusions

The aim of the paper was to verify whether early evaluations concerning FP-like object-oriented metrics based exclusively on programs developed by students hold for open source software as well. For this purpose, the results of the previous work, namely the predictive models that could be derived from the students’ data, were applied to a set of open source programs. The measured data confirmed the first findings, i.e., that FP-like object-oriented metrics can be predicted on the basis of traditional OO metrics (namely, the number of methods), and that OOFPs and CPs seem to be linearly correlated.

Since the analyzed programs are somewhat limited in scope and size, we cannot consider the results reported here as conclusive. We plan to continue our validation activity by applying FP-like metrics to larger object-oriented systems, also in a broader range of domains, in order to seek confirmation of the correlations we found.

References

1. A.J. Albrecht, “Measuring Application Development Productivity”, Proc. Joint SHARE/GUIDE/IBM Application Development Symp., pp. 83-92, 1979.

2. G. Antoniol, C. Lokan, G. Caldiera, R. Fiutem, “A Function Point-Like Measure for Object-Oriented Software”, Empirical Software Engineering, 4 (3): 263-287, 1999.
3. G. Antoniol, C. Lokan, R. Fiutem, “Object-Oriented Function Points: an Empirical Validation”, Empirical Software Engineering, 8 (3): 225-254, September 2003.
4. L. Briand, E. Arisholm, S. Counsell, F. Houdek, and P. Thévenod-Fosse, “Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of The Art and Future Directions,” J. Empirical Software Eng., vol. 4, pp. 387-404, Sept. 1999.
5. S.R. Chidamber and C.F. Kemerer: “A Metric Suite for Object-Oriented Design”, IEEE Trans. Software Eng., vol.20, no.6, pp.476-493, June1994.
6. G. Costagliola, F. Ferrucci, G. Tortora, and G. Vitiello, “Class Point: An Approach for the Size Estimation of Object-Oriented Systems”, IEEE Transactions On Software Engineering, Vol. 31, No. 1, pp. 52-74, January 2005,
7. V. Del Bianco and L. Lavazza, “An Empirical Assessment of Function Point-Like Object-Oriented Metrics”, METRICS 2005, 11th International Software Metrics Symposium, Como, 19-22 September 2005.
8. A.F. Crisà, V. del Bianco, L. Lavazza, “A tool for the measurement, storage, and pre-elaboration of data supporting the release of public datasets”, Workshop on Public Data about Software Development (WoPDaSD 2006), June 10 2006, Como, Italy.
9. H. Kim, C. Boldyreff: “Developing Software Metrics Applicable to UML Models”, 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering, June 11th, 2002.

Introducing Root-Cause Analysis and Orthogonal Defect Classification at Lower CMMI Maturity Levels

Luigi Buglione[1] and Alain Abran[1]

[1] École de Technologie Supérieure - ETS
1100 Notre-Dame Ouest, Montréal, Canada H3C 1K3
Luigi.buglione@computer.org; alain.abran@etsmtl.ca

Abstract. This paper discusses and analyzes possible solutions for achieving an effective process improvement in one specific key process area: measurement, whatever the maturity level and without the constraints of a software process improvement model staged representation. It investigates in particular a Support Process Area, that is, Causal Analysis & Resolution (CAR), together with Orthogonal Defect Classification.

Keywords: Root-Cause Analysis (RCA), CMMI, Capability Level, Orthogonal Defect Classification (ODC), Total Quality Management (TQM).

1 Introduction

Over the past few years in the software engineering community, there has been a growing interest in process improvements and in measurement to support decision-making. Process improvement (PI) can be "measured" and benchmarked against several reference models, and, in the software engineering community, the two best-known benchmarking models are probably:

- **ISO 9001:2000** [28], which is a general domain requirement model that can be instantiated to the software engineering domain using, for instance, the ISO 90003 standard [17];
- **CMMI** (Capability Maturity Model Integration) version 1.2 [11], which is, specifically, a software engineering process model applicable to both the software and systems engineering domains.

While most organizations adopt only one of the two benchmarking models, a few adopt both, and there already exist two-way mappings to compare ISO requirements to CMMI practices for each process.

A few studies have investigated the maturity level equivalence for those organizations already ISO 9001:2000-certified and implementing CMMI or SPICE [22] processes between maturity levels 2 and 3 [24], and they have raised a few issues. For instance, an ISO-certified organization must – to be certified – demonstrate that they have a process in place to identify and eliminate the causes of non conformities (§8.5 – Improvement), implicitly through a Root-Cause Analysis

(RCA). This means that, for these ISO 9001-certified organizations, there should be documented evidence of some measurement-intensive process areas (PAs), such as *Causal Analysis & Resolution (CAR),* which would correspond to the evolution of the Defect Prevention key process area (KPA) at maturity level 5 in the older Sw-CMM [23]) or *Decision Analysis & Resolution (DAR)* at maturity level 3.

However, in a SCAMPI appraisal for the CMMI model [18] adopting a staged evaluation against maturity level 2, these level 3 and level 5 measurement processes would not even be looked into, thereby possibly underrating the real maturity level of such organizations. By contrast, if these organizations were assessed using the continuous representation, this would not be an issue: all processes must then be assessed one by one against the capability process attributes, from level 0 up to level 5, whether or not all the processes connected by a particular maturity level have been implemented [13], as is the case in the staged representation.

Most organizations cannot implement processes at a higher maturity level all at once and from scratch; in practice, they progressively introduce the elements of a higher-level practice by starting with the easiest fit in their own environment, and gradually learning how to master this process, often initially within the limited scope of a pilot project. Only later would these organizations fully deploy a new process, either for business reasons or for a formal maturity assessment. Whatever the kind of model representation chosen (staged or continuous) [2][5][21], it is important to remember why an organization should adopt one or the other. There is no one answer that is valid for every organization: size, number of employees, business type, time-to-market pressure, the business process model adopted and target certifications requested by clients as prerequisites for participating in bids all are examples of some of the parameters to take into account when selecting either a staged or a continuous representation for assessment and benchmarking purposes.

ISO 9001:2000 requires RCA for achieving certification, and it could reasonably be argued that it be positioned within either level 2 or level 3 of CMMI[5]. Positioning the CAR process (a CMMI-related process) at level 5 can be challenged:

Is the CAR practice indeed observed only in organizations with high-level maturity? Could it be introduced at lower maturity levels, and, if so, in what way?

More specifically, can an earlier use of RCA help an organization achieve higher CMMI maturity and capability levels faster?

This paper discusses and analyzes possible strategies for achieving an effective PI by the application of Total Quality Management (TQM) measurement-based tools, whatever the maturity level and the kind of representation (staged or continuous) chosen. This paper focuses in particular on the CAR process of the CMMI Support PA.

Section 2 presents an overview of the role of the Support Processes in PI initiatives. Section 3 presents related work on CAR process and identifies outstanding issues. Section 4 proposes a quantitative usage of CAR as a foundation for achieving higher maturity levels. Section 5 presents a discussion on suggestions concerning the adoption of this quantitative approach to CAR and the benefits of doing so.

[5] According to [24], there is an indicative maturity correspondence between Sw-CMM ML2-3 companies and ISO 9001:1994-certified ones. Taking into account the newer mappings between their respective evolutions (CMMI vs Sw-CMM; ISO 9001:2000 vs 9001:1994), such maturity level equivalence can be assumed.

2 Support Processes in the CMMI

CMMI proposes a classification of process areas (Pas) by typology, grouping them into four classes[6]:

- **Process Management** – includes the cross-project activities related to the defining, planning, deploying, implementing, monitoring, controlling, appraising, measuring and improving processes.
- **Project Management** – includes the project management activities related to planning, monitoring and controlling the project.
- **Engineering** – includes the development and maintenance activities that are shared across engineering disciplines. The engineering PAs were written using general engineering terminology, so that any technical discipline involved in the product development process (e.g. software engineering or mechanical engineering) can use them for process improvement.
- **Support** – includes the activities to support product development and maintenance. The Support PAs address processes that are used in the context of performing other processes. In general, they address processes that are targeted toward the project, and may address processes that apply more generally to the organization.

The Support Processes in the CMMI model are listed in Table 1 – they are divided into Basic and Advanced PAs, and their respective purpose and related General Practices (GPs) are included.

In particular, CMMI states that the Basic Support PAs "*address fundamental support functions that are used by all process areas. Although all Support process areas rely on the other process areas for input, the Basic Support process areas provide support functions that also help implement several generic practices*" while Advanced PAs "*provide the projects and organization with an improved support capability.*"

As can be inferred from Table 1, maturity level 2 Support processes play a dual role in CMMI:

- as **PA**s, and
- as **GP**s

This dual role helps organizations in building the foundations for achieving improvements and contributing to reaching higher maturity levels faster. For instance, a proper Measurement & Analysis (MA) implementation has positive impacts, both on the PAs, PMC and PPQA[7], as well as on the ratings of two GPs, GP3.2 – *Collect Improvement Information* – and GP 4.2 – *Stabilize Subprocess Performance*.

[6] ISO 15504 proposes a similar classification, through the use of five groups, adding a Management (MAN) group. See http://www.isospice.com for details about ISO standard parts and status (parts 1-5 have already been published, and parts 6 & 7 are under development).

[7] A list of acronyms is provided in Appendix A.

Table 1. CMMI v1.2 Support Process Areas (PAs).

Maturity Level	**Process Area – PA**	**Title**	**Process Area Purpose**	**Related General Process - GP**
Basic				
ML2	CM	Configuration Mgmt	Establish and maintain the integrity of work products using configuration identification, configuration control, configuration status accounting and configuration audits	GP2.6
ML2	PPQA	Process & Product Quality Assurance	Provide staff and management with objective insight into processes and associated work products	GP2.9
ML2	MA	Measurement & Analysis	Develop and sustain a measurement capability that is used to support management information needs	GP2.8
Advanced				
ML3	DAR	Decision Analysis & Resolution	Analyze possible decisions using a formal evaluation process that evaluates identified alternatives against established criteria	N.A.
ML5	CAR	Causal Analysis & Resolution	Identify causes of defects and other problems, and take action to prevent them from occurring in the future	GP5.2

3 The Causal Analysis & Resolution (CAR) PA

3.1 CAR Process

CAR is a CMMI process at level 5, and it is expressed by two Specific Goals (SGs) split into five Specific Practices (SPs) – see Table 2.

Table 2. CMMI v1.2 CAR Support Pas

Specific Goals – SG	**Specific Practices – SP**
SG.1	**Determine Cause of Defects**
	SP1.1 - Select defect data for analysis
	SP1.2 – Analyze causes
SG.2	**Address Cause of Defects**
	SP2.1 – Implement the action proposals
	SP2.2 – Evaluate the effect of changes
	SP2.3 – Record data

3.2 Tools for CAR

In particular, SP1.2 sub-practice #2 asks for the following: "*Analyze selected defects and other problems to determine their root causes.*" A recommended analytical tool from TQM [7][8][9] is the Fishbone diagram (or Ishikawa or Cause-Effect diagram) [6]. This quality tool is useful for detecting the root causes of a defect/problem, and for classifying and prioritizing issues in a well-established and ordered manner. In more general terms, as explained in the introduction, the process for detecting and solving problems is usually referred to as RCA (Root-Cause Analysis) in the CMMI practices within the CAR (Causal Analysis & Resolution) process. In Figure 1, an example is presented, where the defect to analyze and remove was "*Software Defects*".

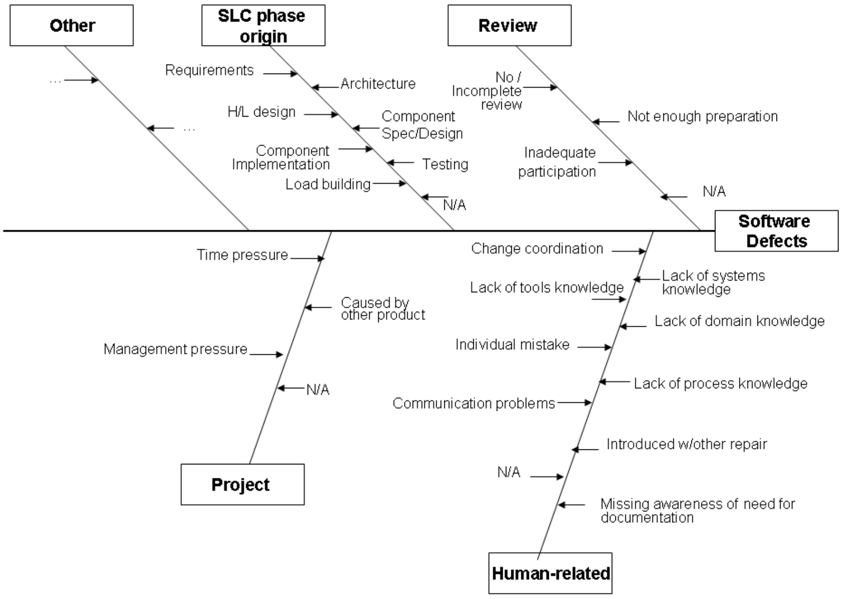


Fig. 1 – *Factors contributing to the high rate of software project failures*[8]

3.3 Related work: Why is CAR positioned at level 5?

The positioning of CAR at level 5 seems to have come about as a result of an assumption by CMMI architects that CAR can be considered as an "evolution" of the "*Defect Prevention*" KPA from the old Sw-CMM, as mentioned in the introduction. On a few occasions, it has been suggested that both RCA and CAR be initially

[8] Elaboration from [15]. Another possible classification for software failures is proposed in [1].

introduced in a qualitative approach at lower maturity levels before including them in a quantitative approach at higher maturity levels:

- Williams (2002) [14] mapped the specific CAR goals against Juran's 10 points, suggesting intensive use of qualitative and quantitative TQM tools for each CAR SP, but without providing suggestions about the "how to" on each tool listed in the fourth point of this list (*Identify root causes*)
- Norausky (2003) [10] proposed a "distributed usage" of CAR across the five maturity levels, using a "*hybrid implementation approach*" for CAR, which would constitute a parallel continuous improvement on CAR implementation, while also pursuing an overall staged representation implementation strategy for those organizations adopting this representation. However, no detailed suggestions are provided for individual maturity levels (from maturity level 1 on), but only suggested usage of high-level CAR measurement.

Thus, a possible solution could be to use CAR at lower maturity levels, and to apply it in a quantitative manner.

4 Quantitative CAR as a foundation for use at Higher Maturity Levels

4.1 From RCA to ODC – Related work

To allow comparability among several instances for a certain problem/effect, Ram Chillarege proposed a technique in the early '90s called Orthogonal Defect Classification (ODC) [3][4], as a way of categorizing defects found both during the development process and after customers receive and begin using the product.

In ODC, defects are classified according to key attributes and then data are analyzed to form the basis for action plans and process improvement activities. ODC is a technique mid-way between the traditional RCA (more qualitative and time-consuming) and Statistical Defect Models (more quantitative, but not easily translatable into corrective action). Through the *orthogonal* classification of defects found (*defect type*) and their association with their *trigger*, it is possible to create consistent and meaningful characterizations of the problems that are found across all software development life cycle stages.

4.2 ODC: Strengths and limitations

A list of strengths and possible limitations in applying ODC has been compiled from the literature:

<u>Strengths</u> [20][25][26] [27]:

- It is an evolution of RCA from a qualitative to a quantitative approach.
- It has adopted a standard taxonomy (types; triggers), which allows comparability across time and organizations.
- It helps in gathering defect data over time, enabling an organization to run statistical analysis and – more generally – to look at defect data in a more objective way.

Limitations:

- It is challenging to use Software Defect Removal, since a large part of the SPI activity is focused on the code. Furthermore, the later a generic defect (not only code) is detected, the more difficult and costly it is to remove [12].
- It is typically applied by organizations having a robust measurement system: ODC needs the capability to consistently gather and analyze data over time; a number of organizations are at lower maturity levels and do not have this capability, or the payback period is too remote for its application to be economical.
- The updating of defect *types* and related defect *triggers* makes it difficult to maintain a backward comparability of source defect data over a long period of time.

4.3 Generalizing and customizing ODC

Our proposed approach is to customize the ODC principles to each implemented PA. In particular, the suggestion is to quantify RCA using the GQM-GQ(I)M [19] approach in the following way: to each low-level leaf (or bone) in a Fishbone diagram, each organization can collect its own groups of causes and adopt this tool whatever its maturity level, and start to do so as early as possible.

Figure 2 illustrates how to determine new, useful measures, or use existing ones, in each related process within a leaf/bone. These measures are shown in blue in Figure 2.

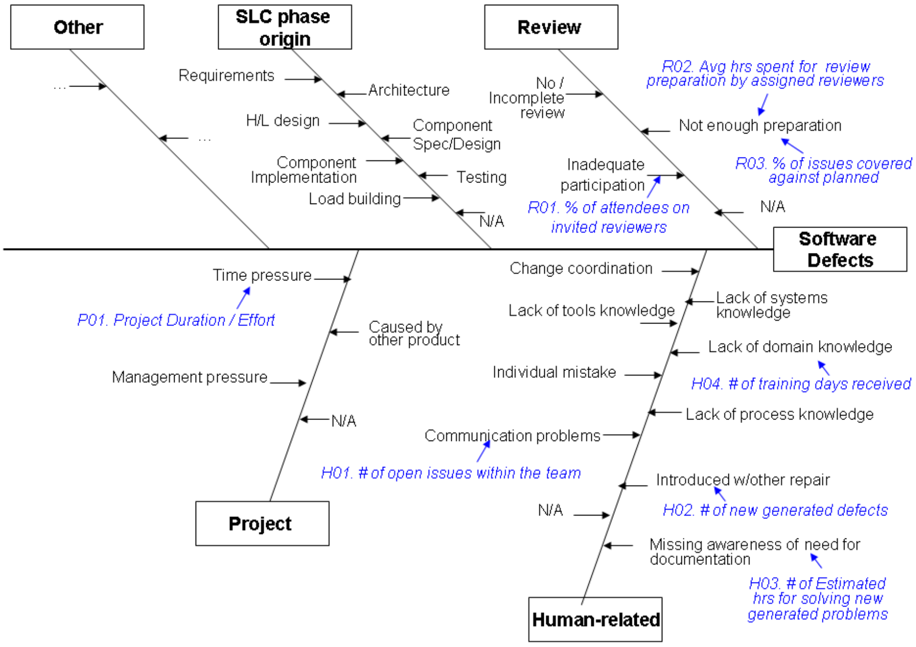

Other
SLC phase origin
Review
...
...
Requirements
Architecture
H/L design
Component Spec/Design
Component Implementation
Testing
Load building
N/A
No / Incomplete review
R02. Avg hrs spent for review preparation by assigned reviewers
Not enough preparation
R03. % of issues covered against planned
Inadequate participation
R01. % of attendees on invited reviewers
N/A
Software Defects
Time pressure
P01. Project Duration / Effort
Caused by other product
Management pressure
N/A
Project
Change coordination
Lack of tools knowledge
Lack of systems knowledge
Lack of domain knowledge
H04. # of training days received
Individual mistake
Lack of process knowledge
Communication problems
H01. # of open issues within the team
Introduced w/other repair
H02. # of new generated defects
N/A
Missing awareness of need for documentation
H03. # of Estimated hrs for solving new generated problems
Human-related

Fig. 2 – *Measures applied to the final bones in a Fishbone diagram*

Some practical guidance is suggested here:

- Build your own defect *types* and *triggers* for each implemented PA – to be refined over time: this will make it possible to define a personalized local standard taxonomy (or start by using RCA classifications such as 3Ms and P (Methods, Machinery, Materials, People) or 4Ps (People, Process, Procedure, Plans).
- Link the measures detected from RCA to their related processes: the suggestion provided by CMMI of gathering only the number and typology of measures[9] seems to be limited to monitoring and controlling CAR. Taking into account more measures derived after the "quantitative" implementation of RCA can help the organization to succeed in the practice of collecting improvement information faster and more easily (GP3.2).

Possible outcomes of such implementation can be:

- Facilitate the adoption of (new) measures needed for removing defects and related causes;
- Facilitate the data collection process in the organization: this is the foundation for statistical analysis later on, at maturity level 4;
- Reduce the cost of non quality (CONQ) in the medium to long term and improve this ratio over the cost of quality (COQ) (e.g. CONQ/COQ ratio): it is reported in [29] that the return on investment from the removal of a cause (these costs are related to CONQ items) is higher than that for removing a defect;

[9] See CAR GP2.8, "Elaboration" section.

- Facilitate the proper implementation of other processes (i.e. Project Monitoring & Control (PMC) and Project and Process Quality Assurance (PPQA)) and general practices such as monitoring and controlling the process (GP2.8), collecting improvement information (GP3.2) and stabilizing sub-process performances (GP4.2) by more skilled resources[10].

5 Discussion

An important goal for every organization is to achieve a real and valuable improvement, with the result that it moves up through the maturity levels. Process improvement models constitute a roadmap for doing so, describing the steps to follow and the techniques and tools to implement. Whatever the model and kind of representation chosen, it is fundamental to properly understand the underlying appraisal principles for the rating process and for deriving useful suggestions for improving processes and related outcomes and outputs.

Measurement and Causal Analysis represent two powerful analytical tools for pursuing PI; they are classified as Support processes by the CMMI model, and should be used and managed in the same way ("dual role" and allocation at a certain maturity level).

Two suggestions for improving the CMMI architecture have been proposed:

- Introduction of the CAR PA area at maturity level 2, as a Basic (rather than an Advanced) Support Process.
    - Supporting rationale: if the software and systems engineering community recognizes Cause-Effect detection and removal ability as a basic process improvement principle, which is also mandatory for ISO 9001 certification and corresponds approximately to CMMI level 2 or 3, then it would be more coherent to classify CAR as a basic process at level 3 than strictly as an advanced process at level 5; this would also improve consistency across both ISO and CMMI benchmarking models.
- Addition of a direct reference to CAR in the general practice related to the capability of adhering to internal processes and policies (GP2.9),: if RCA (and therefore CAR) were to be recognized as a CMMI basic practice, then GP2.9 could be reinforced by introducing a reference to the CAR process[11]. This would help in overcoming the possible risk of maintaining a conservative view of quality (*Quality Assurance*) rather than a proactive one (*Quality Improvement*), which should be at the core of TQM, and therefore of SPI practices.

[10] A less visible, but tangible effect from maturity level 3 on will be to enhance a basic organizational culture of RCA [33][34], introducing and applying it in a gradual manner to each performed process, learning to distinguish, increasingly and at all levels, between defects and causes and their economical impact, and allowing organizations to write their *strategic (process) maps* more and more effectively, as, for instance, in a Balanced Scorecard (BSC).

[11] Each process area can refer to related processes at the end of a certain GP. For instance, GP2.8 refers to the Project Monitoring & Control (PMC) and Measurement & Analysis (MA) processes.

In this paper, we proposed a quantitative approach to RCA and Ishikawa (Fishbone) diagrams, overcoming some limitations noted in the ODC technique, but generalizing some lessons learned from that experience. This approach, going back to TQM studies, would help an organization in its measurement ability, as well as in the rating of other processes at maturity level 2, such as PMC and PPQA. Our suggestion to software organizations is, therefore, to make the teaching of TQM tools a priority in their training programs, not only with reference to CAR, but also as cross-knowledge that would have a positive impact on CMMI (or any other SPI model) processes.

Future work will include the identification of defect *types* and *triggers* for single CMMI PAs, possibly from case studies available in the technical literature, to leverage the advantages of the ODC approach and our suggestion for a quantitative approach to RCA.

References

1. Al-Karaghouli W., AlShawi S. & Elstob M., Enhancing the Understanding of Customer Requirements in Business Software Engineering, Conference on Management Science and Operational Research (ABAS 1999), Barcelona (Spain), URL: http://www.sba.muohio.edu/abas/1999/al-karwa.pdf
2. Buglione, L. & Abran A., Creativity and SPI: an exploratory paper on their measurement, in "Current Trends in Software Measurement", Proceedings of the IWSM 2001 (11th International Workshop on Software Measurement), Montréal, Québec (Canada), August 28-29, 2001, Shaker Verlag, ISBN 3-8265-9681-1, pp. 112-126
3. Chillarege, R., Bhandari, I. S., Chaar, J. K., Halliday, M. J., Moebus, D. S., Ray, B. K. & Wong, M., Orthogonal Defect Classification: Concept for In-Process Measurement, IEEE Transaction on Software Engineering, Vol. 18 No. 11, November 1992, URL: http://www.chillarege.com/odc/articles/odcconcept/odc.html
4. Chillarege, R., ODC – Orthogonal Defect Classification, Presentation, January 19 2000, Center for Software Engineering, IBM Thomas J. Watson Research Center, pp. 92
5. Constant, D., Re: CMMI Representations, which one is the better? Yahoo SPI Mailing List, February 10 2004, URL: http://groups.yahoo.com/group/cmmi_process_improvement
6. Ishikawa, K, Guide to Quality Control (Industrial Engineering & Technology), Asian Productivity Organization; 2/e, 1986, ISBN 9283310357
7. Deming, W. E., Out of the Crisis, MIT Press, 1986, ISBN 0-911379-01-0
8. Juran, J. & Gryna Jr., F. M., Quality Planning and Analysis, McGraw-Hill, 1980, ISBN 0-07-033178-2
9. Crosby, P. B., Quality is free, McGraw-Hill, 1979, ISBN 0-451-62585-4
10. Norausky, G. F., Causal Analysis & Resolution: A Business Driver at all Levels, 2nd Annual Technology and User Group CMMI Conference, NDIA, November 14 2002, URL: http://www.dtic.mil/ndia/2002cmmi/norausky3a1.pdf
11. CMMI Product Team, CMMI for Development, Version 1.2, CMMI-DEV v1.2, Continuous Representation, CMU/SEI-2006-TR-008, Technical Report, Software Engineering Institute, August 2006, URL: http://www.sei.cmu.edu/pub/documents/06.reports/pdf/06tr008.pdf
12. Kan, S. H., Metrics and Models in Software Quality Engineering, 2/e, Addison-Wesley, 2002, ISBN 0201729156

13. Software Engineering Institute, Process Maturity Profile – SCAMPI v1.1 Appraisals Results 2006 Year End Update, Software Engineering Measurement and Analysis Team, Carnegie Mellon University, March 2006, URL: http://www.sei.cmu.edu/appraisal-program/profile/pdf/CMMI/2006marCMMI.pdf
14. Williams, R., Causal Analysis & Resolution (CAR) at Level 1, 3rd Annual Technology and User Group CMMI Conference, NDIA, November 19, 2003, URL: http://www.dtic.mil/ndia/2003CMMI/Slo.ppt
15. Leszak, M., Perry, D. E. & Stoll, D., A Case Study in Root Cause Defect Analysis, Proceedings of the 22nd International Conference on Software Engineering (ICSE 2000), Limerick (Ireland), June 4-11 2000, ISBN 1581132069, pp. 428-437
16. Agena Ltd, Bayesian Belief Nets, URL: http://www.agena.co.uk/bbn_article/bbns.html
17. ISO/IEC, IS 90003:2004 -- Software engineering -- Guidelines for the application of ISO 9001:2000 to computer software, International Organization for Standardization, Geneva, February 2004
18. SEI, Standard CMMI Appraisal Method for Process Improvement (SCAMPI), version 1.1: Method Definition Document, Software Engineering Institute, Handbook, CMU/SEI-2001-HB-001, December 2001, URL: http://www.sei.cmu.edu/pub/documents/01.reports/pdf/01hb001.pdf
19. Tian, J., Quality-Evaluation Models and Measurements, IEEE Software, Vol. 21 No. 3, May/June 2004, pp. 84-91
20. El Emam, K. & Wieczorek, I., The Repeatability of Code Defect Classifications, ISERN Technical Report, ISERN-98-09, Fraunhofer Institute for Experimental Software Engineering, 1998
21. Bate, R., Kuhn, D. A., Wells, C., Armitage, J., Clark, G., Cusick, K., Garcia, S., Hanna, M., Jones, R., Malpass, P., Minnich, I., Pierson, H., Powell, T. & Reichner, A., Systems Engineering Capability Maturity Model (SE-CMM) Version 1.1, Software Engineering Institute, CMU/SEI-1995-MM-003, Maturity Model, November 1995, URL: http://www.sei.cmu.edu/pub/documents/95.reports/pdf/mm003.95.pdf
22. ISO/IEC JTC1/SC7/WG10, TR 15504 – Parts 1-9, Software Process Assessment, v.3.03, 1998
23. Paulk, M. C., Weber, C. V., Garcia, S. M., Chrissis, M. B. & Bush, M., Key Practices of the Capability Maturity Model Version 1.1, Software Engineering Institute/Carnegie Mellon University, CMU/SEI-93-TR-025, February 1993, URL: http://www.sei.cmu.edu/pub/documents/93.reports/pdf/tr25.93.pdf
24. Paulk, M. C., A Comparison of ISO 9001 and the Capability Maturity Model for Software, Software Engineering Institute, CMU/SEI-94-TR-012, July 1994, URL: http://www.sei.cmu.edu/pub/documents/94.reports/pdf/tr12.94.pdf
25. Butcher, M., Munro, H. & Kratschmer, T., Improving software testing via ODC: Three case studies, IBM Systems Journal, Vol. 41, No. 1, 2002, pp. 31-44, URL: http://researchweb.watson.ibm.com/journal/sj/411/butcher.html
26. Dalal, S., Hamada, M., Matthews, P. & Patton G., Using Defect Patterns to Uncover Opportunities for Improvement, Applications of Software Measurement (ASM) Conference, San Jose, CA, February 1999; URL: http://www.argreenhouse.com/papers/paul4/edaP4.pdf
27. Hsiao D., Actionable Root Cause Analysis: Measure for Phase Containment, SEPG2004, Software Engineering Process Group Conference, March 8-11 2003, Orlando, FL (USA)
28. ISO, IS 9001:2000, Quality Management Systems. Requirements, International Organization for Standardization, December 2000
29. Ebert, C., Dumke, R., Bundschuh, M. & Schmietendorf, A., Best Practices in Software Measurement. How to use metrics to improve project and process performance, Springer, 1/e, 2005, ISBN 3-540-20867-4

30. Fredericks, M. & Basili, V., Using Defect Tracking and Analysis to Improve Software Quality, Technical Report, DACS-SOAR-98-2, DoD Data & Analysis Center for Software, 1998, URL: http://www.dacs.dtic.mil/techs/defect/defect.pdf
31. Huber, J. T., A Comparison of IBM's Orthogonal Defect Classification to Hewlett Packard's Defect Origins, Types and Modes, SM /ASM 2000 Conference, March 6-10, 2000, San Jose, CA (USA), URL: http://www.stickyminds.com/stickyfile.asp?i=1764291&j=52901&ext=.pdf
32. Rus, I., Combining Process Simulation and Orthogonal Defect Classification for Improving Software Dependability, ISSRE 2002, Fast Abstract, URL: http://www.chillarege.com/fastabstracts/issre2002/27.pdf
33. Buttles-Valdez, P., People Capability Maturity Model and Anthropology, Software Engineering Institute, 2006, URL: http://www.sei.cmu.edu/sepg/2007/files/Buttles%20Valdez.pdf
34. Howley, P. P., Teaching How to Calibrate a Process Using Experimental Design and Analysis: The Ballistat, *Journal of Statistics Education* Volume 11, Number 2 (2003), URL: www.amstat.org/publications/jse/v11n2/howley.html,

Appendix A: List of Acronyms

BBN	Bayesian Belief Net
BPM	Business Process Model
BSC	Balanced Scorecard
CAR	Causal Analysis & Resolution
CL	Capability Level
CMMI	Capability Maturity Model Integration
DAR	Decision Analysis & Resolution
GP	Generic Practice
ISO	International Organization for Standardization (www.iso.ch)
KPA	Key Process Area
ML	Maturity Level
ODC	Orthogonal Defect Classification
PA	• Process Area (in the CMMI model) • Process Attribute (in the SPICE model)
PI	Process Improvement
PMC	Project Monitoring and Control
PPQA	Process & Product Quality Assurance
RCA	Root-Cause Analysis
SCAMPI	Standard CMMI Appraisal Method for Process Improvement
SP	Specific Practice
SPI	Software Process Improvement
Sw-CMM	Software Capability Maturity Model
TQM	Total Quality Management

Causalities in Software Process Measurement and Improvement

Reiner R. Dumke, Martina Blazey[1], Heike Hegewald[2], Daniel Reitz[3], Karsten Richter[4]

University Magdeburg, Faculty of Computer Science,
Germany
dumke@ivs.cs.uni-magdeburg.de
[1]VW Wolfsburg, Germany, [2]CSC Wonsheim, Germany
[3]EZ T-Systems,Berlin, Germany, [4]Bosch Stuttgart, Germany

Abstract. The following paper characterizes the measurement of software processes considering their modelling, formalization, evaluation and measurement. It shows the current existing experience (rules of thumb, laws, principles etc.) and metrics concept in the software management literature background. Currently many corporations are motivated to achieve the *Capability Maturity Model Integration (CMMI)* level four which includes a quantitative (process) management. Furthermore, the highest CMMI level five includes *causal analysis* between the different process elements/components. Therefore our paper considers different kinds of *causal process networks* in order to qualify the software process mining and reasoning.

Note that our research background of process measurement overview you can find in [5] in our Web site.

1 Introduction

In software processes there are involved many aspects and artefacts in order analyze, managing or control these processes successful. The following figure shows some essential software process involvements defined by Wang and King [27].

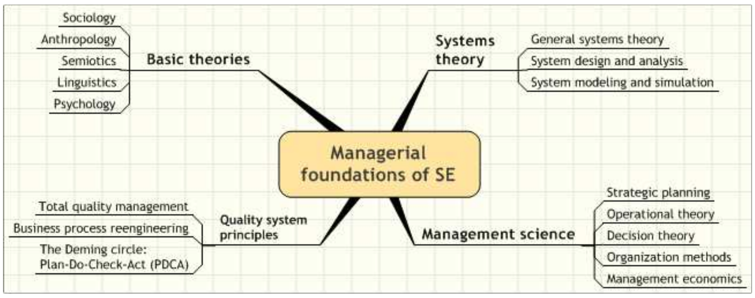


Fig. 1: Managerial foundations of software engineering

The quantitative management as a final goal of the *CMMI level four* in order to keep process controlling is based on the total measurement using process metrics including statistical methods like *Statistical Process Control (SPC)* ([6], [15], [20], [24], [25], [29]).

The *CMMI level five* key process areas – as the next/final step of process improvement based on the CMMI - are

- Organizational innovation and development
- *Causal analysis* and resolution

Therefore, we will consider some kinds of *causality* in software processes based on an overview about approaches of process evaluation and improvement involving process metrics.

2 Causality and Causal Process Networks

Simple examples of semantic networks addressed to the software process are given by [11] and [30] shown in the following charts.

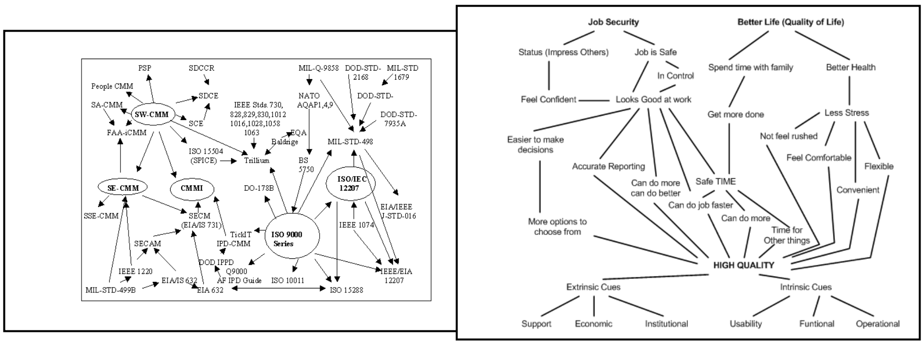


Fig. 2: Examples of semantic networks considering quality models and personal quality indicators

Causal networks as a special kind of semantic networks are very expressive in order to see or analyze the relationships between process activities, areas and indicators in a logical manner. Typical results of such a modelling are ([10], [21])

- The consequence of process activities to other ones involving different quality characteristics like correctness, completeness etc.
- The repercussion of the chosen approaches for process evaluation and improvement

- The overview about strong and weak process connections in order to keep quality improvements
- The application of (causal) model-based principles in order to reduce the process complexity and involvements.

In a general manner a causal network "is a directed acyclic graph arising from an evolution of a substitution system, and representing its history" [28]. The process evolution involves causal relationships between events, states, entities, objects, artefacts etc. which could be based on a special kind of empirical reasoning. In following we will consider special kinds of causal networks in process analysis, measurement, and evaluation. We discuss the following level of *causal process networks* including the kinds of causalities

(D) *"has an influence to"* (e. g. failure propagation)
(I) "*involve an improvement of*" *(*e. g. based on maturity models)
(Q) "*keep the quality of*" (e. g. quality reasoning for connected components).

Note other causalities exists between process components like *"leads to failures in", "decrease the", "has a feedback to", "has side effects from", "involves the ripple effect to"* etc. which are not considered in this paper.

As helpful form of presentation we use the following general schema of software process involvements derived from [4] (see also [3], [16] and [27]).

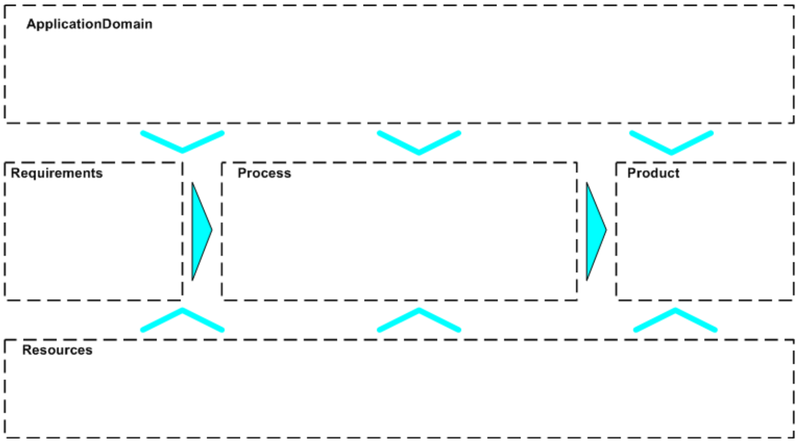


Fig. 3: Software Process Involvements

These process-related areas are the context of our causal process network consideration and investigation explicitly.

3 Causal Process Networks of Dependencies

The dependencies in software processes could be related to anyone and anything. We will start with a first simple example considering the *five core metrics* of Putnam [22]. The definition of the relationships between these metrics leads us to the following simple *causal process network of dependencies*.

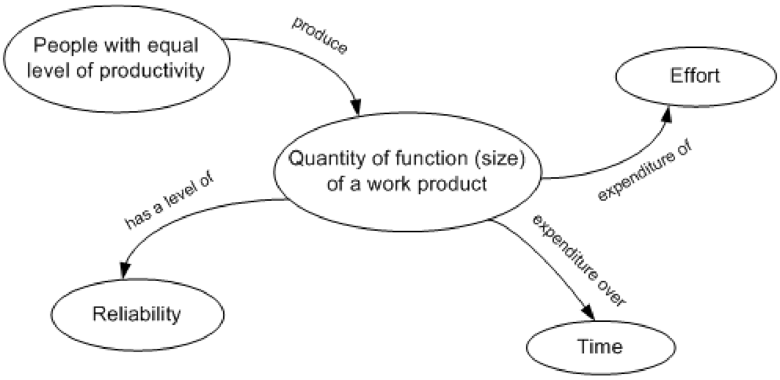

People with equal level of productivity
produce
Effort
Quantity of function (size) of a work product
has a level of
expenditure of
expenditure over
Reliability
Time

Fig. 4: Causal process network of the Five Core Metrics of Putnam and Myers

Using our general process involvements schema, the dependencies will be defined on the basis of [7], [8], [12], [13], [19], [22], [23], [27], and [31]. Adapting this experience can refine the different process aspects shown in the figure 5. We have chosen a rough description and only few of the (causal) relationships in order to keep the clearness of the principles and intentions. The causal process network of dependencies shows

(D1) The existing relationships which are necessary to consider in the different kinds of process aspects managing the changing, migration, upgrading, and evolution

(D2) The different kinds of relationships like *"determines"*, *"requires"*, *"motivates"*, *"forms"*, *"administrates"*, *"performs"* etc. which lead to different kinds causal analysis and causal chain reasoning

(D3) The general network characteristics like direction of the nodes, singularities, and component/chain areas of software process involvements.

Note the shown background of the software process involvements keeps a holistic view of the process dependencies which can be lead o higher level causalities discuss in the following sections.

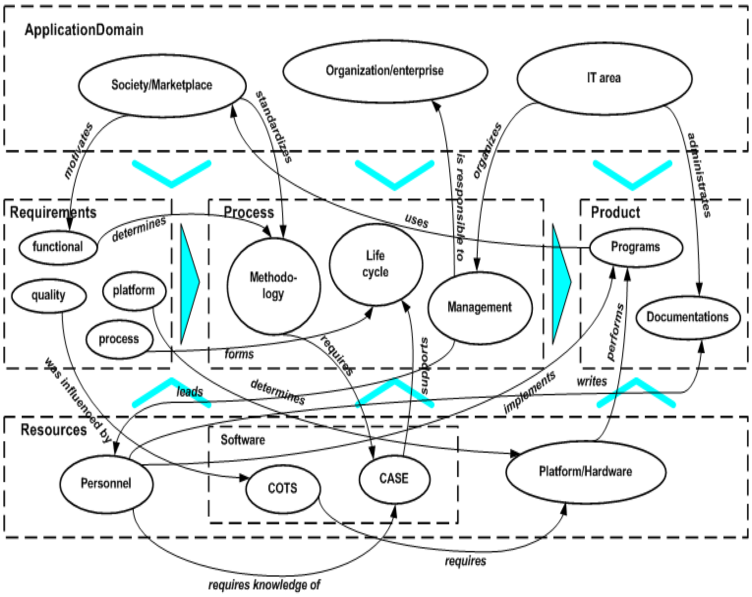


Fig 5: Causal process network of dependencies

4 Causal Process Networks of Improvements

In the next level of causalities in process related networks we will consider the characteristic of improving. We will start again with a very simple *causal process network of improvements* example. The *Software Process Improvement and Capability dEtermination (SPICE)* is defined as an ISO/IEC standard TR 15504 [8]. The principles of the process assessment of SPICE are given in the following semantic network [26] achieving the improvement criteria.

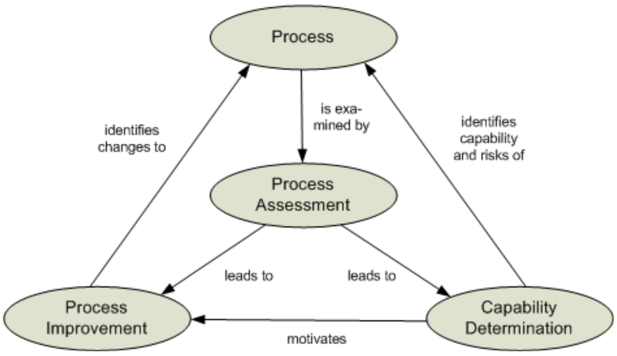


Fig. 6: Causal process network of the SPICE approach

Based on the idea of process improvement, a lot of *maturity models (MM)* were defined and implemented in order to classify different aspects of software products, processes and resources. Some of these maturity evaluation approaches are described in the following table. Their detailed description is given and/or referenced in [1] and [2].

Tab. 1: Chosen maturity models

Model	Description	Model	Description
PEMM	Performance Engineering MM	**CM3**	Configuration Management MM
TMM	Testing Maturity Model	**ACMM**	IT Architecture Capability MM
ITS-CMM	IT Service Capability MM	**OMMM**	Outsourcing Management MM
iCMM	Integrated CMM	**PM2**	Project Management Process Model
TCMM	Trusted CMM	**IMM**	Internet MM
SSE-CMM	System Security Engineering CMM	**IMM**	Information MM
OPM3	Organizational Project Management MM	**PMMM**	Program Management MM
OMM	Operations MM	**PMMM**	Project Management MM
M-CMM	Measurement MM	**IPMM**	Information Process MM
SAMM	Self-Assessment MM	**CPMM**	Change Proficiency MM
UMM	Usability MM	**ASTMM**	Automated Software Testing MM
ECM2	E-Learning CMM	**LM3**	Learning Management MM
WSMM	Web Services MM	**ISM3**	Information Security Management MM
eGMM	e-Government MM	**TMM**	Team MM
EVM3	Earned Value Management MM	**SRE-MM**	Software Reliability Engineering MM
WMM	Website MM	**EDMMM**	Enterprise Data Management MM
DMMM	Data Management MM	**S3MM**	Software Maintenance MM

In order to derive a causal process network of improvements keeping all the process involvements, we will consider the some of the MM’s above that lead us to the following figure.

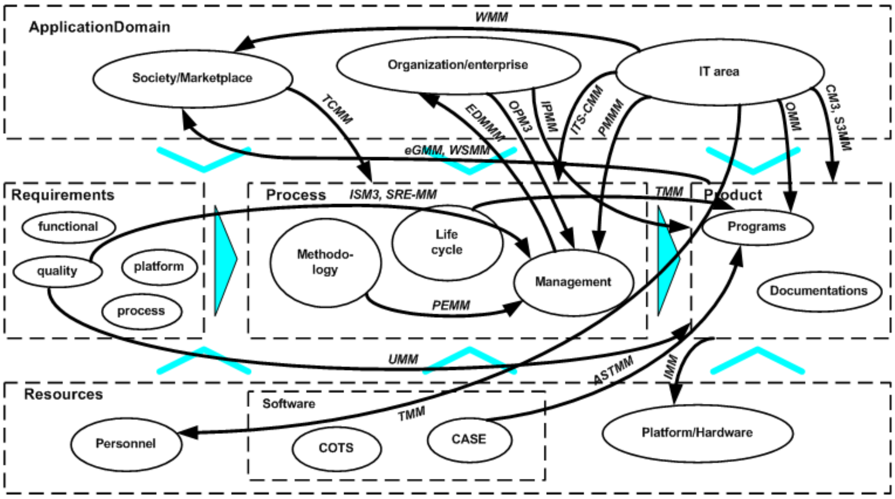


Fig. 7: Causal process network of improvements

The bold arrows visualize the maturity-based improvement characteristics. The causal process network of improvements shows

(I1) The sources and goals of the different improvement models, methods and technologies which could be described in the context of the organizational level

(I2) The components which are not under any activities of (quality) improvements; they could be identified and are the basis for strategic evolvements

(I3) The possibility of adding some attributes which shows the current step of maturity or (quality) level achievements.

Furthermore, improvement activities could also be some techniques or technologies like structured programming, information hiding, coupling reduction, and aspect concerning which we don’t haven consider in this paper.

5 Causal Process Networks of Quality Assurance

Finally, we will discuss the causal process network modelling considering the quality assurance aspects. We start again with a simple kind of *causal process network of quality assurance* based on the *Personal Software Process (PSP)* in the following figure 8 [14].

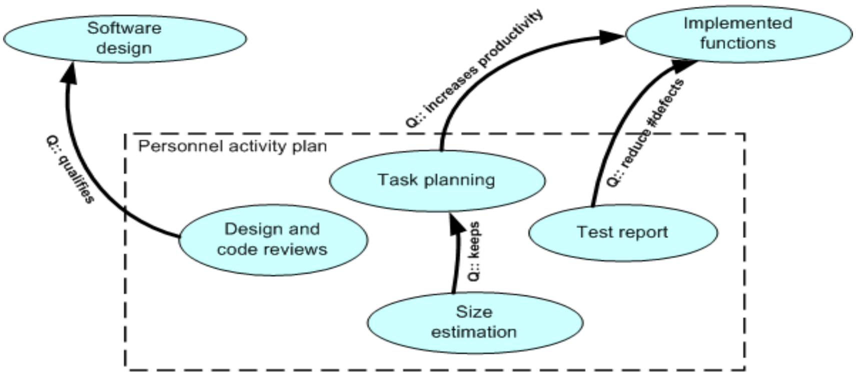


Fig. 8: Causal process network considering PSP

In order to show general quality assurance connections we will select quality approaches like *Six Sigma* (6σ) and *ITIL* (described in [5]) and some of quality principles summarized in [17] as

- Redmill's quality principles in the management of software-based development projects (RP)
- Corbin's methodology for establishing a software development environment (CM)
- Shetty's seven principles of quality leaders (SP)
- Zachmann's quality framework of development complex systems (ZP)
- Kemayel's controllable factors in programmer productivity (KF)

Therefore, we obtain one of the following versions of charts where the kind of quality assurance is given after the Q mark.

The causal process network of quality assurance shows

(Q1) The kinds of involved quality assurance methods and approaches in the current process analysis and reasoning
(Q2) The process involvements including under quality assessment and control in a given context of IT areas
(Q3) The possibilities of network analysis, evaluation, and transformation keeping intended quality levels.

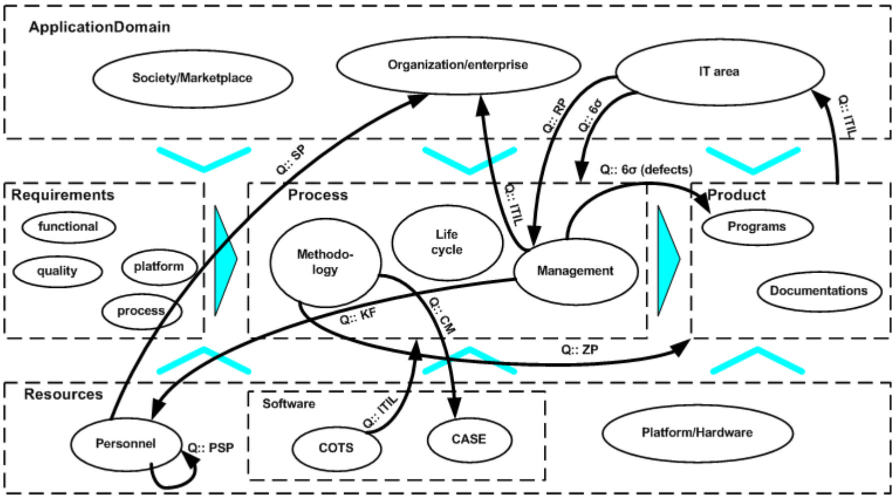


Fig. 9: Causal process network of quality assurance

6 Conclusions and Future Work

Some essential results and open problems are discussed and defined as basics for future investigations in process measurement and evaluation. The chosen paradigm was the causal network as a special kind of semantic networks. In our approach we have discuss the causal analysis and reasoning in a non formal manner in order to simplify the given examples. These are essential aspects in order to keep some of the requirements of the CMMI level five.

Further investigations fulfil the formal background of our approach. On the other hand we will apply this approach in a real industrial environment in order to demonstrate the appropriateness of our approach.

References

1. April, A.: S3m-Model to Evaluate and Improve the Quality of Software Maintenance Process. Shaker Publ., Aachen, Germany 2005
2. Braungarten, R.; Kunz, M.; Dumke, R.: An Approach to Classify Software Measurement Storage Facilities. Preprint No 2, University of Magdeburg, Dept. of Computer Science, 2005
3. Deek, F. P.; McHugh, J. A. M.; Eljabiri, O. M.: Strategic Software Engineering – An Interdisciplinary Approach. Auerbach Publications, Boca Raton London New York,2005
4. Dumke, R.; Braungarten, R.; Blazey, M.; Hegewald, H.; Reitz, D.; Richter, K.: Structuring Software Process Metrics – A holistic semantic network based overview. Proc. of the IWSM 2006, Potsdam, Nov. 2006
5. Dumke, 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. Preprint No 11, University of Magdeburg, 2006 (http://ivs.cs.uni-magdeburg.de/sw-eng/agruppe/froschung/ Preprints.shtml

6. Dumke, R.; Cotè, I.; Andruschak, O.: Statistical Process Control (SPC) – A Metrics-based Point of View of Software Processes Achieving the CMMI Level Four. Preprint No. 7, University of Magdeburg, Fakultät für Informatik, 2004
7. Dumke, R.; Schmietendorf, A.; Zuse, H.: Formal Descriptions of Software Measurement and Evaluation - A Short Overview and Evaluation. Preprint No. 4, Fakultät für Informatik, University of Magdeburg, 2005
8. Emam, K. E.; Drouin, J. N.; Melo, W.: SPICE – The Theory and Practie of Soft-ware Process Improvement and Capability Determination IEEE Computer Society Press, 1998
9. Endres, Albert; Rombach, D.: A Handbook of Software and System Engineering. Pearson Education Limited, 2003
10. Fenton, N,; Krause, P.; Neil, M.: Probalistic Modelling for Softwre Quality Control. Proc. of the European Conference on Symbolic and Quantitative Ap-proaches to Reasoning and Uncertainty, Toulouse 2001
11. Ferguson, J.; Sheard, S.: Leveraging Your CMM Efforts for IEEE/EIA 12207. IEEE Software, September/October 1998, pp. 23-28
12. Garcia, S.: How Standards Enable Adoption of Project Management Practice. IEEE Software, Sept./Oct. 2005, pp. 22-29
13. Haywood, M.: Managing Virtual Teams – Practical Techniques for High-Technology Project Managers. Artech House, Boston, London, 1998
14. Humphrey, W. S.: The Personal Software Process: Status and Trends. IEEE Software, Nov/Dec. 2000, pp. 71-75
15. Juristo, N.; Moreno, A. M.: Basics of Software Engineering Experimentation. Kluwer Academic Publishers, Boston, 2003
16. Kenett, R. S.; Baker, E. R.: Software Process Quality – Management and Control. Marcel Dekker Inc., 1999
17. Keyes, J.: Software Engineering Handbook Auerbach Publ., 2003
18. Kulpa, M. K.; Johnson, K. A.: Interpreting the CMMI – A Process Improvement Approach. CRC Press Company, 2003
19. Lecky-Thompson, G. W.: Corporate Software Project Management. Charles River Media Inc., USA, 2005
20. Pandian, C. R.: Software Metrics – A Guide to Planning, Analysis, and Applica-tion. CRC Press Company, 2004 Pearl, J.: Causality – Models, Reasoning, and Inference. Cambridge University Press, New York, 2000
21. Pearl, J.: *Causality – Models, Reasoning, and Inference.* Cambridge University Press, New York, 2000
22. Putnam, L. H.; Myers, W.: Five Core Metrics – The Intelligence Behind Successful Software Management. Dorset House Publishing, New York, 2003
23. Royce, W.: Software Project Management. Addison-Wesley, 1998
24. SEI: Capability Maturity Model Integration (CMMISM), Version 1.1, Software Engineering Institute, Pittsburgh, March 2002, CMMI-SE/SW/IPPD/SS, V1.1
25. Singpurwalla, N. D.; Wilson, S. P.: Statistical Methods in Software Engineering. Springer Publ., 1999
26. The SPICE Web Site, http://www.sqi.gu.edu.au/spice/ (seen July 24, 2006)
27. Wang, Y.; King, G.: Software Engineering Processes – Principles and Applications. CRC Press, Boca Raton London New York, 2000
28. Weisstein, E.: Causal Networks. Script in Computer Science, http://mathworld.wolfram.com/ CausalNetwork.html (August 1, 2006)
29. Wohlin, C, Runeson, P, Höst, M, Ohlsson, M, Regnell, B, Wesslén, A.: Experimentation in Software Engineering: An Introduction. Kluwer Academic Publishers, Boston, 2000
30. Wong, B. Jefferey, R.: Cognitive Structures of Software Evaluation: A Means-End Chain Analysis of Quality. . In: Bomarius/Komi-Sirviö: Product Focused Software Process

Improvement. PROFES 2001, Kaiserslautern, Sept. 2001, LNCS 2188, Springer Publ., 2001, pp. 6-26

31. Zettel, J.; Maurr, F.; Münch, J.; Wong, L.: LIPE: A Lightweight Process for E-Business Startup Companies Based on Extreme Programming. In: Bomarius/Komi-Sirviö: Product Focused Software Process Improvement. PROFES 2001, Kaiserslautern, Sept. 2001, LNCS 2188, Springer Publ., 2001, pp. 255-270

Software Cost Estimation by Fuzzy Analogy for Web Hypermedia Applications

Ali Idri[1], Azeddine Zahi[2] and Alain Abran[3]

[1] Department of Software Engineering, ENSIAS, Mohamed V University, Rabat, Morocco
E-mail: idri@ensias.ma
[2] Department of Computer Science FST, Sidi Mohamed Ben Abdellah University, Fez, Morocco
E-mail: azahi@fst-usmba.ac.ma
[3] École de Technologie Supérieure, 1180 Notre-Dame Ouest, Montreal, Canada H3C 1K3
E-mail: aabran@ele.etsmtl.ca

Abstract. The aim of this paper is to evaluate the accuracy of Fuzzy Analogy for software cost estimation on a Web software dataset. Fuzzy Analogy is based on reasoning by analogy and fuzzy logic to estimate effort when software projects are described by linguistic values such as *low* and *high*. Linguistic values are represented in the Fuzzy Analogy estimation process with fuzzy sets. However, the descriptions given of the Web software attributes used are insufficient to empirically build their fuzzy representations. Hence, we have suggested the use of the Fuzzy C-Means clustering technique (FCM) and a Real Coded Genetic Algorithm (RCGA) to build these fuzzy representations.

1 Introduction

Software cost estimation has been the subject of intensive investigation in the field of software engineering. As a result, numerous software cost estimation techniques have been proposed and investigated. Software cost estimation by analogy is one of the most attractive techniques and is essentially a form of Case-Based Reasoning [13] [12] [7]. It is based on the following assumption: *similar software projects have similar costs*, and it has been deployed as follows. First, each project is described by a set of attributes that must be relevant and independent. Second, we determine the similarity between the candidate project and each project in the historical database. In the third step, known as case adaptation, the known effort values from the most similar historical projects are used to derive an estimate for the new project. There are two main advantages of analogy-based estimation: first, its process is easy to understand and explain to users; and, second, it can model a complex set of relationships between the dependent variable (such as cost or effort) and the independent variables (cost drivers). However, its deployment in software cost estimation still warrants some improvements. Hence, we have developed a new approach referred to as Fuzzy Analogy based on reasoning by analogy and fuzzy logic to estimate effort when software projects are described by linguistic values [7], which is a major limitation of all estimation techniques (categorical data: nominal or ordinal scale) such as ‘very low’, ‘low’ and ‘high’. Indeed, handling imprecision, uncertainty and partial truth is unavoidable when using these values. As a consequence, Fuzzy Analogy suggests the use of fuzzy sets rather than classical intervals or numbers (as in the classical procedure of cost estimation by analogy) to represent linguistic values. The main motivation behind the theory of fuzzy sets, founded by Zadeh in 1965, is the desire to build a formal quantitative framework that captures the vagueness of human knowledge, since it is expressed via natural language [15].

In an earlier work, we validated Fuzzy Analogy on the COCOMO’81 dataset that contains 63 historical software projects [3]; each of which is described by 12 attributes measured on an ordinal scale composed of six linguistic values: ‘very low’, ‘low’, ‘nominal’, ‘high’, ‘very high’ and ‘extra high’ [7]. The fuzzy representation (fuzzy sets and their membership functions) of the 12 COCOMO’81 cost drivers has been empirically achieved based on their descriptions [5]. The accuracy of Fuzzy Analogy is compared with that of three other models: classical analogy, Intermediate COCOMO’81 and fuzzy Intermediate COCOMO’81. Fuzzy Analogy performs better in terms of accuracy (MMRE=21) and in its adequacy in dealing with linguistic values [7].

The aim of this work is to validate Fuzzy Analogy on a dataset containing 54 Web hypermedia applications [10]. Each application is described by 9 numerical attributes, such as the number of html or shtml files used, the number of media files and team experience (Table 1). Initially, this dataset contains more than 9 software attributes, but some of them may be grouped together. For example, we have 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 number of Web pages in the application (Webpages).

Table 1. Software attributes for the Web dataset

Software attribute	Description
Teamexp	Average number of years’ experience the team has in web development
Devteam	Number of people who have worked on the software project
Webpages	Number of Web pages in the software
TextP	Number of text pages in the software (600 words to a text page)
Imag	Number of images in the software
Anim	Number of animations in the software
Audio/video	Number of audio/video files
Tot-high	Number of high-effort features
Tot-nhigh	Number of low-effort features

The validation of Fuzzy Analogy on the Web dataset requires the determination of the fuzzy sets, and their membership functions, associated with the 9 Web software attributes. However, the descriptions given of these attributes are, unlike the case of COCOMO’81, insufficient to empirically build their fuzzy representations. Hence, we have suggested the use of the Fuzzy C-Means clustering technique (FCM) and a Real Coded Genetic Algorithm (RCGA) to build fuzzy representations for software attributes [8]. So, we apply the FCM-RCGA process to the 9 Web software attributes.

2 Fuzzy Analogy: An Overview

Fuzzy Analogy is a ‘fuzzification’ of the classical analogy procedure. It is also composed of three steps: identification of cases, retrieval of similar cases and case adaptation [7]:

▪ *Identification of a Case***:** The goal of this step is the characterization of all software projects by a set of attributes. Each software project is described by a set of selected attributes that are measured by linguistic values. Let us assume that we have M attributes, and, for each attribute V_j, a measure with linguistic values is defined (A_k^j). Each linguistic value A_k^j is represented by a fuzzy set with a membership function ($\mu_{A_k^j}$). The fuzzy sets and their membership functions are defined by using: 1) empirical techniques which construct membership functions from expert knowledge; or 2) automatic techniques, which construct membership functions from historical data using clustering techniques.

▪ *Retrieval of Similar Cases*: This step is based on the choice of a software project similarity measure. This choice is very important, since it will influence which analogies are found. We have proposed a set of candidate measures for software project similarity [6]. These measures evaluate the overall similarity $d(P_1, P_2)$ of two projects P_1 and P_2, by combining the individual similarities of P_1 and P_2 associated with the various attributes V_j describing P_1 and P_2, $d_{v_j}(P_1, P_2)$.

$$d(P_1, P_2) = \begin{cases} \textit{all of } (d_{v_j}(P_1, P_2)) \\ \textit{most of } (d_{v_j}(P_1, P_2)) \\ \textit{many of } (d_{v_j}(P_1, P_2)) \\ \ldots. \\ \textit{there exists of } (d_{v_j}(P_1, P_2)) \end{cases} \quad \textbf{(1)}$$

To evaluate the overall distance of P_1 and P_2, the individual distances $d_{v_j}(P_1, P_2)$ are aggregated using Regular Increasing Monotone (RIM) linguistic quantifiers such as ‘all’, ‘most’, ‘many’, ‘at most α’, or ‘there exists’. The choice of the appropriate RIM linguistic quantifier, Q, depends on the characteristics and needs of each environment. Q indicates the proportion of individual distances that we feel is necessary for a good evaluation of the overall distance.

▪ *Case Adaptation:* The objective of this step is to derive an estimate for the new project by using the known effort values of similar projects. In this step, two issues must be addressed. First, the choice of how many similar projects should be used in the adaptation, and, second, how to adapt the chosen analogies to generate an estimate for the new project. In Fuzzy Analogy, we have proposed a new strategy for selecting projects to be used in the adaptation step. This strategy is based on the distances $d(P, P_i)$ and the definition adopted in the studied environment for the proposition, ‘ P_i is a closely similar project to P .’ For the adaptation formula, the weighted mean of all known effort projects in the data set is used.

3 Building fuzzy sets and their membership functions for the Web software attributes

The use of Fuzzy Analogy to estimate software development effort requires determination of the fuzzy sets, and their membership functions, of the attributes describing software projects. Because the descriptions given of the 9 Web software attributes are insufficient to empirically build their fuzzy representations, we suggest the use of the Fuzzy C-Means clustering technique (FCM) and a Real Coded Genetic Algorithm (RCGA) to build the fuzzy representations of the Web software attributes.

The proposed FCM-RCGA fuzzy set generation process consists of two main steps (Figure 1). First, we use the well-known FCM algorithm and the Xie-Beni validity criterion to decide on the number of clusters (fuzzy sets) [2] [14]. Second, we use an RCGA to build membership functions for these fuzzy sets [4] [11]. Membership functions can be trapezoidal, triangular or Gaussian.

The FCM algorithm is a fuzzy clustering method used to generate a known number of clusters (c) from a set of numerical data $\mathrm{X}=\{\mathrm{x}_1,...,\mathrm{x}_\mathrm{n}\}$. The determination of this number is still an open problem in clustering. Often, empirical knowledge or a set of evaluation criteria is used to choose the best set of clusters. In this work, we use the fuzzy cluster validity criterion proposed in [14]. FCM is an iterative algorithm that aims to find cluster centers $(C_j), 1\le j\le c$ and the matrix $U=(u_{ij}), 1\le i\le n, 1\le j\le c$ that minimizes the following objective function:

$$Min\ J_p(U,C)=\sum_{i=1}^{i=n}\sum_{j=1}^{j=c}(u_{ij})^m\left\|x_i-c_j\right\|^2 \quad \text{subject to} \quad \sum_{i=1}^{c}u_{ij}=1, \forall j=1,...,n \qquad \mathbf{(2)}$$

where m is the control parameter of fuzziness; $U=(u_{ij})$ is the partition matrix, containing the membership values of all data in all clusters;

After generating fuzzy sets (clusters $(C_j), 1\le j\le c$) with their partition $U=(u_{ij})$ by means of FCM, we use an RCGA to build membership functions for these clusters [4] [11]; membership functions can be trapezoidal, triangular or Gaussian. Our RCGA consists in building a set of membership functions (μ_j), $1\le j\le c$ that interpolates and minimizes the mean square error, which is defined as follows:

$$MSE(\mu_1,..,\mu_c)=\frac{1}{n}\sum_{j=1}^{j=n}\left\|(\mu_1(x_j),..,\mu_c(x_j))-(u_{1j},..,u_{c,j})\right\|^2 \qquad \mathbf{(3)}$$

subject to $\sum_{j=1}^{c}\mu_j(x_i)=1$, for all x_i and $\mu_j(x_i)=u_{ij}$, $1\le i\le n$; $1\le j\le c$

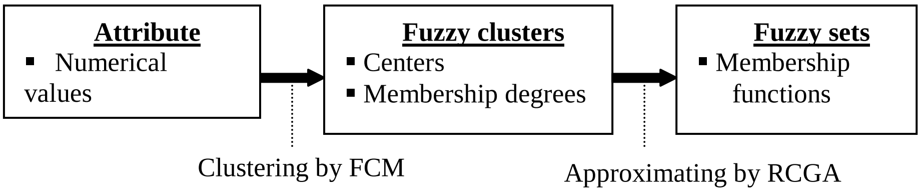


Fig. 1. Fuzzy set generation process

The use of an RCGA to find membership functions μ_j requires the determination of certain parameters, such as the coding scheme, the fitness function and the various genetic operators (selection, crossover and mutation). Concerning the coding scheme, a chromosome in the population of our RCGA, $m_i, 1 \leq i \leq M$, represents the set of the unknown membership functions, $(\mu_j), 1 \leq j \leq c$, associated with the c fuzzy sets generated by the FCM. The shape of the membership functions can be trapezoidal, triangular or Gaussian. Thus, each chromosome encodes a set of membership functions in a real vector $(m_i^1,...,m_i^K)$. The genes m_i^j are obtained from the shape of the membership functions. Figure 2 shows the structure of a chromosome, *m*m_i, encoding trapezoidal membership functions. The fitness function F is obtained using the following formula:

$$F(m_i) = \frac{MSE(m_i)}{\sum_{i=1}^{i=M} MSE(m_i)} \; ; MSE(m_i) = \frac{1}{n} \sum_{j=1}^{j=n} \left\| \mu(x_j) - y_j \right\|^2 \qquad \textbf{(4)}$$

where $\mu(x_j) = (\mu_1(x_j),...,\mu_c(x_j))$, $y_j = (u_{j1},...,u_{jc})$ and M is the size of the population. For the three genetic operators (selection, crossover and mutation), we use those that are specific to RCGAs. Hence, the linear ranking is used as a selection operator [1]. The line recombination method is considered as a crossover operator [11]. The Breeder Genetic Algorithm is used as a mutation operator [11].

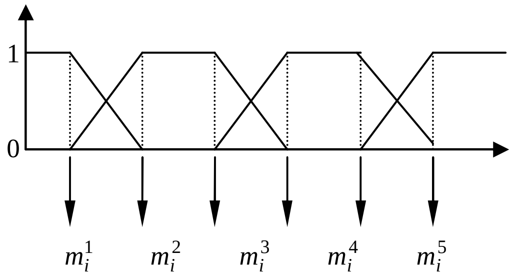


Fig. 2. Structure of a chromosome associated with trapezoidal membership functions

4 Empirical Results

This section presents and discusses the results obtained when applying Fuzzy analogy to the Web dataset. The calculations were made using a software prototype developed with Matlab 7.0 under a Microsoft Windows PC environment. The accuracy of the estimates is evaluated by using the magnitude of relative error, MRE, defined as:

$$MRE = \left| \frac{Effort_{actual} - Effort_{estimated}}{Effort_{actual}} \right| \quad (5)$$

The MRE is calculated for each project in the dataset. In addition, we use the measure prediction level *Pred*. This measure is often used in the literature. It is defined by:

$$\Pr ed(p) = \frac{k}{N} \quad (6)$$

where N is the total number of observations, k is the number of observations with an MRE less than or equal to p. In this evaluation, we use p equal to 0.20. The *Pred(0.20)* gives the percentage of projects that were predicted with an MRE less than or equal to 0.20.

Normally, for the overall distances, each environment must define its appropriate quantifier (Q) by studying its features and its requirements. Because a lack of knowledge concerning the appropriate quantifier for the environment from which Web dataset was collected, we used various α-RIM linguistic quantifiers to combine the individual similarities. An α-RIM linguistic quantifier is defined by a fuzzy set in the unit interval with membership function *Q* given by:

$$Q(r) = r^{\alpha} \quad \alpha > 0 \quad (7)$$

For each Web software attribute, several experiments were conducted with the FCM algorithm, each time using a different initial matrix U. The desired number of clusters (c) is varied within the interval [2,7]. The parameter m is fixed to 2 in all experiments. As mentioned earlier, we use the Xie-Beni criterion to decide on the number of clusters. For each attribute, we choose the number of clusters that minimizes the value of the Xie-Beni criterion. Table 2 shows the 'best' classification obtained according to the Xie-Beni index.

Table 2: Example of a classification generated by the FCM algorithm.

Attributes	#fuzzy sets	Attributes	#fuzzy sets	Attributes	#fuzzy sets
DevTeam	7	TEXTP	3	Audio	4
Teamexp	7	IMAG	3	Tot-high	5
Webpages	2	ANIM	3	Tot-nhigh	5

After generating the fuzzy sets with the FCM Algorithm, we applied the RCGA algorithm, as designed in the previous section, to these fuzzy clusters to build their membership functions. This algorithm is applied with populations of up to 300, the mutation probability fixed to 0.9, and the number of generations is equal to 200. Figure 3 shows three different shapes of membership functions associated with the fuzzy sets of the IMAG and TEXTP attributes respectively.

By analyzing the results of the validation of the Fuzzy Analogy technique (Table 3 and Figure 4), we noted that the accuracy of the estimates depends on the linguistic quantifier (α) used in the evaluation of the overall similarity between software projects. So, if we consider the accuracy measured by *Pred*(0.20) as a function of α, we can say that, in general, it increases monotonously according to α. This is because our similarity measures decrease monotonously according to α. Indeed, when α tends towards zero, this implies that the overall similarity will take into account fewer attributes among all those describing Web software projects. The minimum number of attributes to consider is one. As a consequence, the overall similarity will be higher because we are more likely to find at least one attribute in the Web dataset for which the associated linguistic values are the same for the two projects. By contrast, when α tends towards infinity, it implies that the overall similarity will take into account many attributes among all the available ones describing the software projects. As a maximum, we may consider all attributes. Consequently, the overall similarity will be minor because we are more likely to find one attribute in the Web dataset for which the associated linguistic values are different for the two projects.

We compared the accuracy of the Fuzzy Analogy when using different shapes of the membership functions. Our findings were the following: Fuzzy Analogy performs better when trapezoidal membership functions are used than when triangular membership functions are used. In addition, it performs better when triangular membership functions are used than when Gaussian membership functions are used. However, the accuracy of Fuzzy Analogy is higher for the three shapes of membership functions than that of threshold precision, which is often used in the software cost estimation literature (Pred(20)≥70).

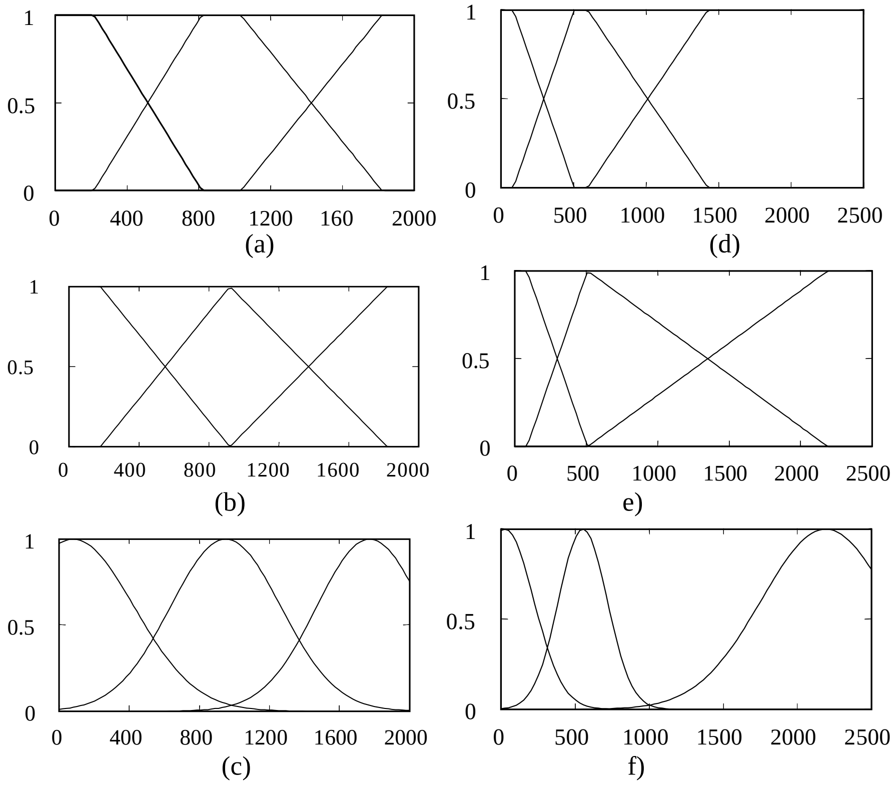


Fig. 3. Membership functions associated with the fuzzy sets of the IMAG and TEXTP attributes respectively: (a) Trapezoidal for IMAG; (b) Triangular for IMAG; (c) Gaussian for IMAG; (e) Trapezoidal for TEXTP; (e) Triangular for TEXTP; and (f) Gaussian for TEXTP.

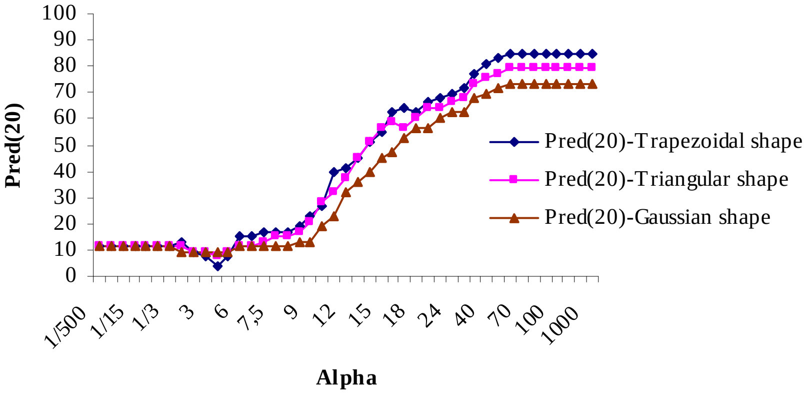


Fig. 4. Relationship between α and the accuracy of Fuzzy Analogy for the three shapes of membership functions.

Table 3. Results of the evaluation of Fuzzy Analogy

	Trapezoidal functions		Trinagulair functions		Gaussian functions	
alpha-RIM	Pred(0.20)	MMRE	Pred(0.20)	MMRE	Pred(0.20)	MMRE
1/10	11.32	717.78	11.32	718.90	11.32	719.54
1/7	11.32	711.12	11.32	712.67	11.32	713.53
1/3	11.32	683.10	11.32	686.37	11.32	688.12
1	13.21	600.72	11.32	607.91	9.43	612.33
3	7.55	434.93	9.43	445.37	9.43	455.48
7	16.98	250.73	13.21	260.14	11.32	272.85
10	26.42	177.59	28.30	184.28	18.87	195.68
15	54.72	119.31	56.60	122.88	45.28	131.48
25	71.70	78.36	67.92	81.12	62.26	87.93
30	77.36	69.91	73.58	73.17	67.92	79.75
40	81.13	62.19	75.47	65.75	69.81	72.40
50	83.02	59.71	77.36	63.30	71.70	70.17
60	84.91	58.92	79.25	62.54	73.58	69.46
70	84.91	58.68	79.25	62.30	73.58	69.28
80	84.91	58.60	79.25	62.23	73.58	69.22

5 Conclusion and future work

In this paper, we have validated the Fuzzy Analogy approach for estimating the cost of Web hypermedia applications. The use of Fuzzy Analogy requires the determination of the fuzzy sets, and their membership functions, of the attributes describing these applications. Because the descriptions given of the Web software attributes are insufficient to empirically build their fuzzy representations, we have suggested the use of the Fuzzy C-Means clustering technique (FCM) and a Real Coded Genetic Algorithm (RCGA) (the FCM-RCGA process) to build these representations for the Web software attributes. The membership functions generated may be trapezoidal, triangular or Gaussian. The results of this validation show that Fuzzy Analogy generates accurate estimates, using trapezoidal, triangular or Gaussian fuzzy representation. It should be noted that our findings favor the use of trapezoidal representation.

We are currently looking at comparing Fuzzy Analogy and classical analogy on the Web dataset, as we have already compared Fuzzy Analogy with three other models (Intermediate COCOMO’81, Fuzzy Intermediate COCOMO’81 and classical analogy) on the COCOMO’81 dataset. We have found that Fuzzy Analogy performs better in terms of accuracy and its adequacy in dealing with linguistic values. Another interesting avenue of research would be to look at the accuracy of Fuzzy Analogy when using the FCM-RCGA process rather than empirical knowledge for building fuzzy sets.

Acknowledgments

We thank Dr. Emilia Mendes for providing us with the data on the 54 Web software projects.

5 Bibliography

1. Baker, J. E. Reducing bias and inefficiency in the selection algorithm. Lawrence Erlbaum Associates, Publishers, ProcICGA 2 (1987), 14-21.
2. Bezdek, J., Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum, New York, (1981).
3. Boehm, B. W., Software Engineering Economics. Prentice-Hall, (1981).
4. Herrera, F., Loazano, M., Sanchez, A. M., A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study. Int. J. Intell. Syst. 18(3), (2003) 309-338.
5. Idri, A., Kjiri, L., Abran, A., COCOMO Cost Model Using Fuzzy Logic. Proceedings of the 7th International Conference on Fuzzy Set Theory and Technology, NJ, (2000) 219-223.
6. Idri, A., Abran, A., A Fuzzy Logic-faced Measure for Software Project Similarity: Validation and Possible Improvements. 7th IEEE International Symposium on Software , 4-6 April, London, (2001) 85-96.
7. Idri, A., Abran, A., Khoshgoftaar, T., Estimating Software Project Effort by Analogy Based on Linguistic Values. 8th IEEE International Software Metrics Symposium, Ottawa, June (2002) 21-30.
8. Idri, A., Zahi, A., Abran, A., Generating Fuzzy Term Sets for Software Project Attributes using Fuzzy C-Means and Real Coded Genetic Algorithms, to be published at ICT4M, Malaysia, November 2006.
9. Medasani, S., Kim, J., Krishnapuram, R., An overview of membership function generation techniques for pattern recognition. Internat. J. Approx. Reas., 19 (1998) 391-417.
10. Mendes, E., Triggs, W. C., Mosley, N., Counsell, S., A comparison of Development Effort Estimation Techniques for Web Hypermedia Applications. 8th IEEE International Software Metrics Symposium, Ottawa (2002), 131-140.

11 Mühlenbein, H., Schlierkamp-Voosen, D. Predictive Models for the Breeder Genetic Algorithm: I. Continuous Parameter Optimization," *Evolutionary Computation*, Vol. 1, No. 1 (1993), 25-49.

12 Shepperd, M., Schofield, C., Estimating Software Project Effort Using Analogies. IEEE Transactions on Software Engineering, vol. 23, no. 12 (1997), 736-747.

13. Vicinanza, S., Prietolla, M. J., Case-Based Reasoning in Software Effort Estimation. Proceedings of the 11th Int. Conf. on Information Systems (1990).
14. Xie, X. L., Beni, G., "A validity measure for fuzzy clustering." IEEE Transactions on Pattern Analysis Machine Intelligence, vol. 13, no 8 (1991), 841-847.
15. Zadeh, L. A., Fuzzy sets. Information and Control, vol. 8 (1965), 338-352.

Re-Assessing the Intention to Use a Measurement Procedure based on COSMIC-FFP[*]

Nelly Condori-Fernández[1], Oscar Pastor[1],

[1] Department of Information Systems and Computation
Valencia University of Technology, Spain
{nelly, opastor}@dsic.upv.es

Abstract. This paper describes the replication of an empirical study that was designed to evaluate the adoption of RmFFP in practice. RmFFP is a measurement procedure designed to measure the functional size of object-oriented systems from requirements specifications obtained in the context of the OO-Method approach. This procedure has been designed in accordance with the COSMIC-FFP standard method. The evaluation is based on the Method Adoption Model (MAM), where the intention to use a method is determined by the users' perceptions. The results show that an intention to use RmFFP exists, and that it is more influenced by usefulness than ease of use.

1 Introduction

Functional size measurement (FSM) methods currently play a crucial role in software project management, IFPUG Function Point Analysis (FPA) being the most popular. However, a rapid evolution of development paradigms has given rise to a new FSM method, COSMIC-Full Function Point (FFP) [1], which is more compatible with modern software engineering concepts and is applicable to various software domains. COSMIC-FFP has been awarded the ISO/IEC19761 standard [2] and is seen as the first second-generation FSM method.

In recent years we have been working on a method based on model transformation called the OO-Method [3], supported by an automatic code generation tool, Oliva Nova [4]. This tool includes a module that allows the estimating of the functional size of applications from conceptual models (object model, dynamic model, and functional model) in function points [5]. However, this estimation process is at present carried out during the analysis phase of the OO-Method development process, and our intention is to enrich the Oliva Nova tool by allowing estimation of functional size at an earlier stage using high-level specifications.

To implement this we have designed an FSM procedure called RmFFP, based on the COSMIC-FFP standard method for estimating the functional size of object-oriented systems generated by the OO-Method from requirements specifications [6]. We have applied this procedure to various case studies (Rent a Car Management, Golf Management, and Maintenance Service Management) with groups of undergraduate

[*] Supported by the DESTINO project, ref. TIN2004-03534, Spain

students to evaluate its reproducibility and productivity [7]. However, there is a need also to assess users' response to the new procedure and their intention to use it in the future, for which reason we have designed an empirical study, which has been carried out twice. For the first evaluation, we used computer science major students as experimental subjects. However, for the second evaluation, we used PhD students in Computer Science; this last evaluation is the subject-matter of this paper.

The empirical study designed is based on a pre-existing theoretical model called the Method Adoption Model (MAM) [8]. This model includes the same primary constructs as the Technology Acceptance Model [9], which has been adapted to explain and predict the adoption of methods. These constructs are:

- *Perceived Ease of Use:* the extent to which a person believes that using a particular method would be effort-free.
- *Perceived Usefulness:* the extent to which a person believes that a particular method will be effective in achieving the intended objectives.
- *Intention to Use:* the extent to which a person intends to use a particular method.

This empirical study has been carried out to evaluate the intention to use RmFFP on the basis of the MAM constructs, which was also applied by Poels [10] y Abrahao [11].

This paper is organized as follows: Section 2 presents an overview of the evaluation process carried out. Section 3 describes the assessment of the intention to use RmFFP in the future. Section 4 discusses the analysis and interpretation of these results. Finally, Section 5 sets out our conclusions and indicates further work to be carried out.

2 General Description

The process of evaluation carried out was initiated with the selection of participants, who formed part of a training process. The aim of the training was to develop a level of expertise required for the subjects to be able to measure requirements specifications using the RmFFP measurement procedure. As shown in Figure 1, a package of training materials was utilized, which was designed and prepared in advance, comprising specific case studies with the OO-Method Requirements Model [12], examples of the use of RmFFP and a measurement guide.

At the end of the training process, the participants demonstrate what they have learned by means of the measurement of a case study selected in advance. If this demonstration is not satisfactory, a training session will be given again for reinforcement. If satisfactory, the next step will be the recording of perceptions and intentions of the participants on the use of RmFFP by means of a questionnaire, which is described in more detail in Section 3. Each of the answers of the participants is then recorded, validated and analyzed, to be finally interpreted and presented in a report.

This evaluation process was carried out twice. The results obtained in the first evaluation were reported in [13]. However, we replicated the empirical study with the purpose of improving the reliability of the results.

The experimental planning carried out is described below.

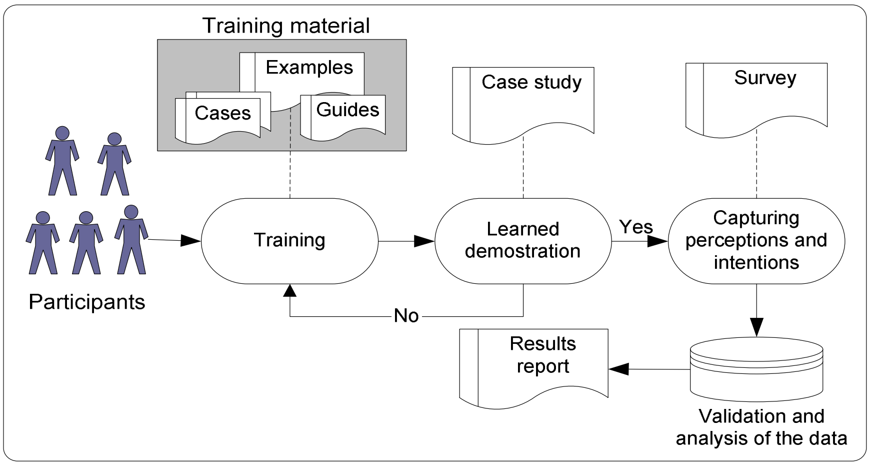


Fig. 1. General process carried out to evaluate the intention to use RmFFP in the future

3 Re-assessing the intention to use RmFFP

According to the Goal/Question/Metric template [14], the goal of the empirical study was to analyze user's responses for the purpose of assessing RmFFP with respect to its intention to use from the viewpoint of researchers in the context of students measuring OO-Method requirements specifications.

3.1 Experiment Planning

The **subjects** were eleven Computer Science PhD students at the Valencia University of Technology, enrolled in the "*Software Technologies*" course during the period from February to June of 2006. All subjects were familiar with modelling and measuring techniques.

The **independent variable** is the variable for which the effects should be evaluated. In our study, this variable corresponded to the functional size measurement procedure RmFFP. The **dependent variables** were the three perception-based variables of the MAM: *perceived ease of use (PEOU), perceived usefulness (PU), and intention to use (ITU).*

We wanted to test the following **hypotheses**:

- *H_1:* There is an intention to use RmFFP.
- *H_2:* Intention to use is determined by perceived ease of use and perceived usefulness.

From Hypothesis H2 we can derive two simple hypotheses, which are the following:

- *H_3:* Intention to use is determined by perceived ease of use.
- *H_4:* Intention to use is determined by perceived usefulness.

The **instruments** used in this experiment included the experimental object, training materials and a survey. The *experimental object* was the OO-Method requirements specification of three case studies: a Car Rental application, the Management of a Maintenance Service in a Hospital, and Golf Management. The *training materials* were the following: a set of instructional slides on the OO-Method Requirements Model and the RmFFP procedure; a case-study that describes an example of the application of RmFFP, a measurement guide, and another case study to verify the training.

The original *survey* was adjusted for the replication of this empirical study. Table 1 summarizes the main changes carried out.

Table 1. Differences between the original survey and the adjusted survey

Original survey, adapted from [5]	Adjusted survey
- **PEOU construct included 5 items: I1, I3, I4, I6 and I9.** - **PU construct included 5 items: I2, I5, I8, I10, and I11.** - **IU construct included 3 items: I7, I12, and I13.** - **Items I2 and I11 were moved to evaluate the PEOU construct as items I2 and I12 respectively.** - **Items I12 were moved to evaluate the PEOU construct as items I14.**	- PEOU construct included 8 items: I1, I2, I3, I4, I6, I9, I12 and I14. - PU construct included 3 items: I5, I8, and I11. - IU construct included 4 items: I7, I10', I13', and I15. - Items I10' and I13' were increased to evaluate the IU construct. - Items I11 and I15 are items I10 and I13 (in the original survey) respectively.

This adjusted survey[12] included fifteen closed questions, based on the items used to measure PEOU, PU, and ITU, as shown in Figure 2. The items were formulated using a 5-point Likert scale, using the opposing statement question format.

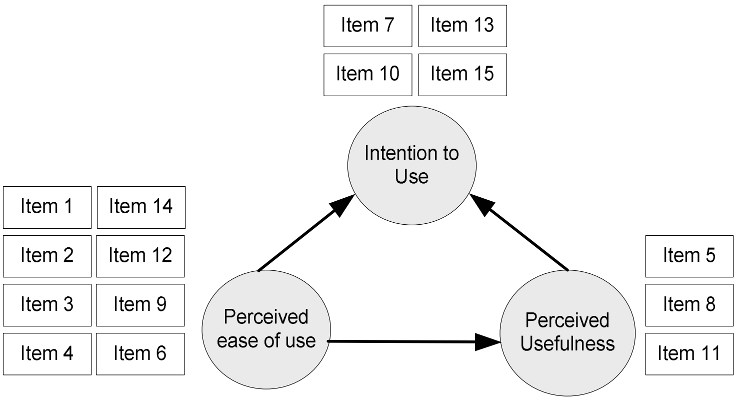


Fig. 2. Operationalized Method Adoption Model

[12] http://www.dsic.upv.es/~nelly/survey2.pdf

3.2 Experiment Operation

When the participants demonstrated what they had learned by means of the measurement of an OO-Method requirements specification, no time-limit was set and the interaction among subjects was controlled to avoid plagiarism.

Once the training phase had ended, we carried out a ***post-task survey***, in which the subjects were asked to complete a survey to evaluate their perceptions of RmFFP use and intentions of use.

3.3 Validity evaluation

In order to ensure that the experimental results are valid, we considered the following threat to **construct validity**:

Inadequate pre-operational explanation of constructs: This threat means that the constructs are not sufficiently defined, and hence the experiment cannot be sufficiently clear. We used an **inter-item correlation analysis** to evaluate the construct validity of the variables PEOU, PU, and ITU. We employed two criteria, Convergent Validity (CV) and Discriminant Validity (DV), for each item; if convergent validity was higher than discriminant validity, the item would be validated. However, we found that the CV value was lower than the DV value for items I2 and I12 (see Table 6 of the appendix). For this reason these two items were removed from the analysis.

In addition, we also conducted a **reliability analysis** on the validated items to calculate the degree to which the values of the constructs are free of measurement error. The reliability analysis was conducted using the Chronbach alpha technique, where the corresponding alpha value for each MAM construct is shown in Table 2.

Table 2. Reliability analysis for the MAM constructs

Construct	Cronbach(α)	α without I2 and I12
PEOU	0.71	0.802
PU	0.818	0.818
ITU	0.846	0.846

These values indicate that the items included in the survey are reliable, alphas of 0.7 or above being acceptable according to Nunally [15]. In addition, the alphas of PU and ITU were better than the alphas obtained in the first evaluation (PU = 0.5 and ITU = 0.5), more details in [13].

4 Analysis and Interpretation

Once the data were collected and validated (see Table 5 of the appendix), the scores of each subject were averaged over the different items that are relevant for a construct, and we obtained three mean values for each subject. Table 3 shows descriptive statistics for each construct of the MAM; we note that the mean ITU score obtained with eleven subjects is greater than 3.

Table 3. Descriptive statistics for PEOU, PU and ITU

Statistic	PEOU	PU	ITU
Mean	3.98	3.67	3.61
Standard dev.	0.59	0.87	0.89
Minimum	2.83	2.33	1.75
Maximum	5.00	5.00	5.00

To evaluate the intention to use RmFFP, hypothesis H1 was formally tested by verifying whether the scores that the students assigned to this construct were significantly better than the middle score on the 5-point Likert scale. We verified the normality of these data using the Shapiro-Wilk test. As the data distribution was normal, we used the one-tailed sample t-test to evaluate the statistical significance of the observed differences in mean ITU. The objective was to verify whether the scores that students assign to the construct of the MAM were significantly higher than the score of 3. The statistical test was applied with a significance level of 5%, i.e. alpha = 0.05. The results of the t-tests (Table 4) allow rejection of the null hypotheses with medium significance level, meaning that we empirically corroborated the intention to use RmFFP in the future.

Table 4. One Sample t-test for Intention to Use variable

Statistic	ITU
Mean Difference	.614
95% Conf. Interval for the diff.	.016 (lower) 1.212 (upper)
t	2.29
1-tailed p-value	.022

In order to test hypothesis H2, regression analysis technique was applied. The regression equation resulting was: **ITU=-0.18+0.76*PU+ 0.25*PEOU**.

The regression model had a medium significant level (p = 0.0133), which means that H2 was confirmed. The determination coefficient (R^2 = 0.66) indicated that 66% of total variation in intention to use can be explained by variation in the perceived usefulness and perceived ease of use.

With respect to Hypothesis H_3: Perceived ease of use → Intention to use. The regression equation resulting from the analysis is: **ITU = 1.151 + 0.618* PEOU**. The regression had a low significance level (p = 0.212), which means that H3 was not confirmed.

Finally, with respect to Hypothesis H_4: Perceived usefulness → Intention to use. The regression equation resulting from the analysis is: **ITU = 0.621 + 0.816* PU**. The regression had a high significance level (p = 0.003), which means that H4 was confirmed. The determination coefficient (r2 = 0.635) showed that 63.5% of the total variation in intention to use can be explained by variation in perceived usefulness. Figure 3 represents the lineal regression obtained.

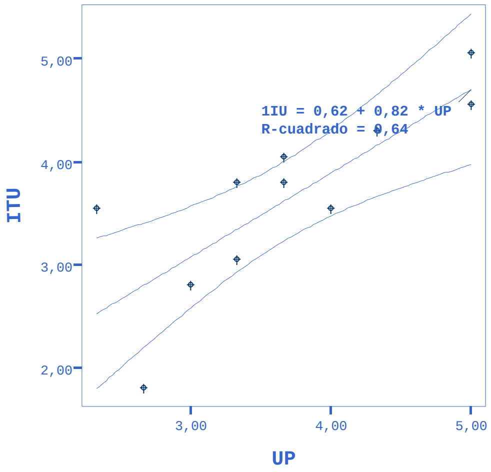


Fig. 3. Regression model: Intention to Use vs. Usefulness Perceived

5 Conclusions

This paper describes the replication of an empirical study that evaluates the intention to use the RmFFP procedure, which was designed for measuring the size of object oriented systems, in accordance with the COSMIC-FFP method. The results indicate that there is an intention to use RmFFP when sizing OO-Method requirement specifications. Although RmFFP is perceived as easy to use, the results of our tests show that perceived usefulness can have a stronger influence on intention to use RmFFP than perceived ease of use. This means that the user intends to use RmFFP more because of its usefulness, in terms of its accuracy in estimating other indicators, than because of its ease of use.

Therefore, the MAM relation between the Perceived Ease of Use and the Intention to Use could not be verified empirically in the software measurement domain.

In a future study, we plan to identify and evaluate other variables that may affect the intention to use a measurement procedure.

Acknowledgements

We would like to thank Alain Abran for their helpful comments and suggestions.

References

1. Abran A., J. M. Desharnais, S. Oligny, D. St-Pierre, and C. Symons, "COSMIC-FFP Measurement Manual Version 2.2, The COSMIC Implementation Guide for ISO/IEC 19761:2003", École of technologie supérieure- ETS, Montreal (Canada) 2003. Available free at: www.gelog.etsmtl.ca/cosmic-ffp
2. ISO, ISO/IEC 19761 Software Engineering-COSMIC-FFP-A Functional Size Measurement Method, International Organization for Standardization_ISO, Geneva, 2003.

3. Pastor O., Gomez J., Insfran E., Pelechano V., 2001. “The OO-Method approach for information systems modelling: from object-oriented conceptual modelling to automated programming”, Information Systems 26, pp. 507-534.
4. CARE S.A: http://www.care-t.com/ , last visited 15 September 2006
5. Abrahão S., "On the Functional Size Measurement of Object-Oriented Conceptual Schemas: Design and Evaluation Issues", PhD Thesis, Department of Information Systems and Computation, Valencia University of Technology, October 2004.
6. N. Condori-Fernández, S. Abrahão, O. Pastor, Towards a Functional Size Measure for Object-Oriented Systems from Requirements Specifications. IEEE Quality Software Int. Conf. 2004, Germany, pp. 94-101.
7. N. Condori-Fernández, O. Pastor, Evaluating the Productivity and Reproducibility of a Measurement Procedure, ER Workshop on Quality of Information Systems, Minnesota, USA, November 2006.
8. Moody D. L., "Dealing with Complexity: A Practical Method for Representing Large Entity Relationship Models", PhD. Thesis, Department of Information Systems, University of Melbourne, Australia, 2001.
9. F. D. Davis, “Perceived Usefulness, Perceived Ease of Use and User Acceptance of Information Technology”, *MIS Quarterly*, 1989, pp. 319-340.
10. Poels G., Maes A., Gailly F., Paemeleire R., “Measuring User Beliefs and Attitudes towards Conceptual Schemas: Tentative Factor and Structural Equation Model”, Fourth Annual Workshop on HCI Research in MIS, December 2005.
11. Abrahao S., Poels G., Pastor O. “A Functional Size Measurement Method for Object-Oriented Conceptual Schemas: Design and Evaluation Issues”. Software & System Modelling, Springer Verlag, 5(1): 48-71, 2005.
12. Insfran E., Pastor O. and Wieringa R., “Requirements Engineering-Based Conceptual Modelling”. Journal Requirements Engineering, Springer-Verlag, 7(2): 61-72, 2002.
13. N. Condori-Fernández, O. Pastor, “An Empirical Study on the Likelihood of Adoption in Practice of a Size Measurement Procedure for Requirements Specification” IEEE Quality Software International Conference, China, October 2006.
14. Wohlin C., Runeson P., Höst M., M. C. Ohlsson, B. Regnell, and A. Wesslén, Experimentation in Software Engineering: An Introduction, 2000.
15. Nunally J., Psychometric Theory, McGraw-Hill, 2nd ed., New York, NY1978.

Appendix

Table 5. Data set used in the analysis

Students	I1	I2	I3	I4	I6	I9	I12	I14	I5	I8	I11	I7	I10	I13	I15
1	3	3	5	5	3	4	2	5	4	2	2	1	1	3	2
2	5	4	5	3	2	5	4	5	4	3	4	5	2	5	4
3	4	4	4	3	4	4	3	5	4	3	2	3	3	4	1
4	4	4	5	4	3	4	4	5	5	4	4	4	4	5	4
5	4	4	5	5	5	4	3	5	4	3	3	4	4	3	4
6	4	3	4	3	4	4	4	4	3	4	4	4	4	4	3
7	3	4	4	3	3	3	4	4	4	4	4	3	4	4	3
8	5	3	5	5	5	5	5	5	5	5	5	5	5	5	5
9	4	5	4	3	4	4	5	5	5	5	5	5	4	5	4
10	2	4	3	2	3	3	3	4	4	3	3	3	3	3	3
11	4	4	4	4	4	2	4	4	4	2	1	3	4	4	3

Table 6. Correlation between Survey Items (Construct Validity)

		PEOU								PU			ITU				Mean		Valid
		I1	I2	I3	I4	I6	I9	I12	I14	I5	I8	I11	I7	I10	I13	I15	CV	DV	
PEOU	I1	1,00	-0,07	0,63	0,36	0,28	0,61	0,56	0,52	0,26	0,32	0,33	0,69	0,30	0,71	0,47	0,49	0,44	YES
	I02	-0,07	1,00	-0,31	-0,44	-0,13	-0,26	0,27	0,09	0,38	0,15	0,09	0,31	0,13	0,24	0,08	0,02	0,20	NO
	I03	0,63	-0,31	1,00	0,79	0,07	0,63	0,01	0,72	0,31	0,03	0,18	0,18	-0,11	0,29	0,39	0,44	0,18	YES
	I04	0,36	-0,44	0,79	1,00	0,48	0,25	-0,12	0,49	0,28	-0,11	-0,12	-0,12	0,07	-0,07	0,27	0,35	0,03	YES
	I06	0,28	-0,13	0,07	0,48	1,00	0,03	0,23	0,12	0,13	0,29	0,04	0,23	0,65	-0,08	0,20	0,26	0,21	YES
	I09	0,61	-0,26	0,63	0,25	0,03	1,00	0,18	0,74	0,26	0,43	0,60	0,50	-0,11	0,44	0,37	0,40	0,35	YES
	I12	0,56	0,27	0,01	-0,12	0,23	0,18	1,00	-0,02	0,47	0,79	0,70	0,82	0,72	0,83	0,68	0,27	0,71	NO
	I14	0,52	0,09	0,72	0,49	0,12	0,74	-0,02	1,00	0,57	0,16	0,22	0,25	-0,21	0,33	0,20	0,46	0,22	YES
PU	I05	0,26	0,38	0,31	0,28	0,13	0,26	0,47	0,57	1,00	0,49	0,42	0,38	0,31	0,56	0,52	0,64	0,37	YES
	I08	0,32	0,15	0,03	-0,11	0,29	0,43	0,79	0,16	0,49	1,00	0,91	0,71	0,66	0,64	0,58	0,80	0,39	YES
	I11	0,33	0,09	0,18	-0,12	0,04	0,60	0,70	0,22	0,42	0,91	1,00	0,74	0,43	0,62	0,70	0,78	0,38	YES
ITU	I07	0,69	0,31	0,18	-0,12	0,23	0,50	0,82	0,25	0,38	0,71	0,74	1,00	0,57	0,73	0,76	0,77	0,43	YES
	I10	0,30	0,13	-0,11	0,07	0,65	-0,11	0,72	-0,21	0,31	0,66	0,43	0,57	1,00	0,38	0,53	0,62	0,26	YES
	I13	0,71	0,24	0,29	-0,07	-0,08	0,44	0,83	0,33	0,56	0,64	0,62	0,73	0,38	1,00	0,52	0,66	0,41	YES
	I15	0,47	0,08	0,39	0,27	0,20	0,37	0,68	0,20	0,52	0,58	0,70	0,76	0,53	0,52	1,00	0,70	0,41	YES

Predicting Quality Attributes via Machine-Learning Algorithms

Hakim Lounis[1], M.K. Abdi[2], H. Yazid[1]

[1] Department of Computer Science, Université du Québec à Montréal, CP 8888, succ. Centre-Ville, Montréal, QC, H3C 3P8, Canada
lounis.hakim@uqam.ca, HOURIA.YAZID.1@ens.etsmtl.ca

[2] Department of Computer Science and Operations Research, Université de Montréal, CP 6128, succ. Centre-Ville, Montréal, QC, H3C 3J7, Canada
abdimust@iro.umontreal.ca

Abstract. Software metrics provide quantitative means to control the software development and the quality of software products. Getting a set of valid and useful metrics is not only a matter of definition; the entire process includes, among other steps, theoretical and empirical validation of theses metrics to assure their utility. This work is about empirical validation of object-oriented metrics via machine learning algorithms; it aims at empirically verify the relationships between object-oriented design decisions and three quality attributes: change impact, fault-proneness, and, reusability. Several algorithms, belonging to various machine learning approaches, are selected and run on software data collected from medium size applications.

Keywords: Software product quality, change impact, fault-proneness, reusability, machine-learning.

1 Introduction

The big issue in software engineering is to produce high quality software in time and in budget. To pass through this issue, measurement based on metrics should be done at different stages of the software development life cycle, with objectives as, software quality prediction or learning from past experiences. In fact, software metrics provide a quantitative approach to control the software development and then the software products quality. They are also a crucial source of information for decision-making. A large number of object-oriented (OO) measures have been proposed in the literature [1] [2] [3]. A particular emphasis has been given to the measurement of design artifacts, in order to help assess quality early on during the development process. Such measures are aimed at providing ways of assessing the quality of software. Such an assessment of design quality is objective, and the measurement can be automated. But how do we know what measures actually capture important quality aspects? As it is stated by the ISO/IEC international standard (14598), internal metrics are especially helpful when they are related to external quality attributes, e.g., maintainability, reusability, usability, etc. In the present work, we explore the relationship between three quality factors (or external attributes), change impact, fault proneness, and, reusability, (the two former are related to maintainability) from one side, and internal attributes as inheritance, coupling, cohesion, complexity, and size, from the other side. The aim is to try to predict these factors via metrics associated to the considered internal attributes. Many different approaches have been proposed to build such empirical predictive models; for example, they can be mathematical models (case of statistical techniques like linear and logistic regression) or artificial intelligence-based models (case of machine-learning techniques). In all cases, they allow affecting a value to a quality characteristic based on the values of a set of software measures, and they allow the detection of design and implementation anomalies early in the software life cycle. They also allow organizations that purchase software to better evaluate and compare the offers they receive.

Our work deals then with the construction of such predictive models for the three pre-cited quality factors. To build these models, different Machine-Learning (ML) algorithms are selected. We first present in section 2 previous works in the same topic. We then introduce in section 3 the hypotheses we want to verify with selected ML algorithms. These algorithms are presented in section 4. This section describes also the empirical process we follow and provides the experimental results we have obtained. Finally, conclusions, current research, and directions for future research are outlined.

2 Related work

Several studies were conducted to validate metrics and to relate them to some properties. Li and Henry [3] took five metrics of Chidamber and Kemerer [2], added three of their own, to show that there is a strong relationship between these metrics and maintenance effort, expressed in number of changed lines code. In [4], the authors showed that the choice of architectures, in early stages of software systems design, has an important impact on a number of quality factors, for instance, maintainability, efficiency, and reusability. In [5], Lounis & al. proposed a succession of 24 code metrics to generate predictive models related to fault- proneness. On the other hand, in [6], the authors also studied relationships between most of the coupling metrics, cohesion, and inheritance ones in one side, and classes' fault-proneness in the other side. In [7], Mao & al. worked to predict, starting from internal measures, the reusability degree, which was measured, by the necessary amount of work to reuse a component either from a system to another one in the same domain. Less works have been conducted on change impact. Han [8] developed an approach for computing change impact on design and implementation documents. In [9], Antoniol & al. predicted evolving object-oriented systems size starting from the analysis of the classes impacted by a change request. They predicted changes size in terms of added/modified lines of code. In an other work, Kung and al [10], interested by regression testing, developed a change impact model based on three links: inheritance, association, and, aggregation. They also defined formal algorithms to calculate all the impacted classes including ripple effects. Li and Offutt examined in [11], the effects of encapsulation, inheritance, and polymorphism on change impact; they also proposed algorithms for calculating the complete impact of changes made in a given class. On the other hand, Briand and al., in [12], tried to see if coupling measures, capturing all kinds of collaboration between classes, can help to analyze change impact. Lastly, in [13] and [14], a change impact model was defined at an abstract level, to study the changeability of object-oriented systems. The adopted approach uses characteristic properties of object-oriented systems design, measured by metrics, to predict changeability.

Many of the measures proposed and their relationships to external quality attributes have been the focus of little empirical investigation. However, different approaches have been proposed to build such empirical assessment models. Most of the work has been done using statistical techniques [3] [15]. As far as we know, Porter and Selby [16] have been the first to use a ML algorithm to automatically construct software quality models. They have used a classification algorithm, to identify those product measures that are the best predictors of interface errors likely to be encountered during maintenance. After Selby & Porter, many others, e.g., [17], [18], have used classification algorithms to construct software quality predictive models. More recently, [19] have investigated ML algorithms with regard to their capabilities to accurately assess the correctability of faulty software components; an inductive logic programming system, presented the best results from the point of view of model accuracy. On the other hand, in most above-mentioned techniques, the estimation process depends on threshold values that are derived from a sample base. This dependency raises the problem of representativity for the samples, as these often do not reflect the variety of real-life systems. Thus, in a previous study, we also worked on the identification of trends instead of the determination of specific thresholds by replacing them with fuzzy thresholds [20].

The following section introduces the hypotheses we have verified with the ML techniques.

3 Object-oriented design hypotheses

We are interested in exploring the relationship between three external quality attributes (quality factors), change impact, fault proneness, and reusability, from one side, and internal attributes as inheritance, coupling, cohesion, complexity, and size, on the other side.

3.1 Maintainability expressed as change impact

We are interested by the relationship between inter-classes dependencies, namely coupling, an architectural property, and change impact. Our objective is to see which types of coupling influences more change impact. In our domain literature review, we have noticed that very few works propose a more or less complete definition of change impact, i.e., model taking into account main links that one can find in an object-oriented design (namely, association, aggregation, invocation and inheritance). In our study, we took the impact model defined in the SPOOL project [21]; we consider it as one of the most general. It allows impact calculation in a systematic way; this is an important factor concerning effort and maintenance cost reduction.

When a change is considered, it is necessary to identify system components that will be impacted; it will ensure that the system will still run correctly after change implementation. Our concern is then focused on how the system reacts to a change (in general). It is generally accepted that a system absorbs easily a change if the number of impacted components is small. A component refers to a class, a method, or a variable. As examples of changes, one can have the deletion of a variable, the change in a method's scope from "public" to "protected" or the removal of the relationship between a class and its parent. A total of 13 changes are identified. In this context, we call change impact the set of classes that require a correction after this change. In our work, we are interested only in changes that have a syntactic impact; a given change is characterized by a code transformation somewhere in the system. If the system is successfully re-compiled, then there is no impact; otherwise, we have an impact. In [22], Abdi & al. gives the final list containing a total of 52 changes, including 12 changes for variables, 25 for methods, and, 15 for classes.

On the other hand, we worked with a set of metrics related to coupling. They are presented in table 1.

Table 1. Coupling metrics.

Metrics	Definition
RFC	Response For a Class: number of methods called upon in response to a message.
MPC	Message Passing Coupling: number of messages sent by a class in direction of the other classes of the system.
CBOU	CBO Using: refers to the classes used by the target class.
CBOIUB	CBO Is Used By: refers to the classes using the target class.
CBO	Coupling Between Object: number of classes with which a class is coupled.
CBONA	CBO No Ancestors: CBO without considering the classes ancestors.
AMMIC	Ancestors Method–Method Import Coupling: number of parents classes with which a class has an interaction of the method-method type and a coupling of the type IC.
OMMIC	Others Method–Method Import Coupling: number of classes (others that super classes and subclasses) with which a class has an interaction of the method-method type and a coupling of the type IC.
DMMEC	Descendants Method–Method Export Coupling: number of subclasses with which a class has an interaction of the method-method type and a coupling of the type EC.
OMMEC	Others Method–Method Export Coupling: number of classes (others that super classes and subclasses) with which a class has an interaction of the method-method type and a coupling of the type EC.

3.2 Maintainability expressed as components fault-proneness

The goal of this hypothesis statement is to empirically investigate the relationship between object-oriented design measures and fault-proneness at the class level. We therefore need to select a suitable and practical measure of fault-proneness as the dependent variable for our study. For instance, in a non-faulty component, there was not any change of corrective type and, in a faulty one, there were one or more changes of corrective type during the development/maintenance phase.

The measures of coupling, cohesion, and inheritance identified in a literature survey on object-oriented design measures are the independent variables used in this study. 28 coupling metrics are used; Briand & al. [1] proposed a comprehensive suite of metrics to measure the level of class coupling during the design of OO systems. The set of metrics quantifies the different OO design mechanisms (for instance, friendship between classes, specialization, and aggregation). 18 different types of coupling are derived as listed in table 2.

Table 2. More design coupling metrics.

Symbol	Metrics
IFCAIC	Inverse Friend CA Import Coupling
ACAIC	Ancestors CA Import Coupling
OCAIC	Others CA Import Coupling
FCAEC	Friend CA Export Coupling
DCAEC	Descendant CA Export Coupling
OCAEC	Others CA Export Coupling
IFCMIC	Inverse Friend CM Import Coupling
ACMIC	Ancestors CM Import Coupling
OCMIC	Others CM Import Coupling
FCMEC	Friend CM Export Coupling
DCMEC	Descendant CM Export Coupling
OCMEC	Others CM Export Coupling
IFMMIC	Inverse Friend MM Import Coupling
AMMIC	Ancestors MM Import Coupling
OMMIC	Others MM Import Coupling
FMMEC	Friend MM Export Coupling
DMMEC	Descendant MM Export Coupling
OMMEC	Others MM Export Coupling

The 10 others are: coupling between object (CBO, and CBO’), response for class (RFC_$_1$, and RFC_α), Message passing coupling (MPC), Information-flow-based coupling (ICP, IH-ICP, and NIH-ICP), and Data abstraction coupling (DAC, and, DAC’). We considered also 10 cohesion metrics: LCOM$_i$ (Lack of cohesion of methods, i=1..5), Co and Coh (LCOM variants), LCC (Loose Class Cohesion), TCC (Tight Class Cohesion), and, ICH (Information-flow-based Cohesion). More details about all these coupling and cohesion metrics could be found in [23]. Finally, the selected inheritance metrics are 11: DIT (depth of inheritance tree), HIT *(*height of inheritance tree), NOA (number of ancestors), NOC (number of children), NMI (number of inherited methods), NOP (number of polymorphic methods), AID (average inheritance depth), NOD (number of descendants), NMO (number of overridden methods), NMA (number of methods added, new methods), and SIX (specialization index). We focus on design measurement since we want the measurement-based models investigated in this study to be usable at early stages of software development. Furthermore, we only use measures defined at the class level since this is also the granularity at which the fault data could realistically be collected.

3.3 Components reusability

Our concern is to verify the relationship between reusability at the class level, and some internal object-oriented attributes. The basic idea is that some internal attributes like inheritance, coupling, complexity, or size can be good indicators of the possibility of reuse of a component. Establishing a direct relation between these attributes (which are measurable) and the potential of a component to be reusable (which is not directly measurable) can be an interesting track for the automation of detection of potentially reusable components.

Reusability is a complex factor, which is domain dependent. Some components are more reusable in one domain than in others. Our goal is not to search for a set of methods measuring reusability universally but to study some specific aspects and characteristics pertaining to OO programming languages, e.g. C++ that affect reusability. Different aspects can be considered to measure empirically the reusability; one aspect is the amount of work needed to reuse a component from a version of a system to another version of the same system. Another aspect is the amount of work needed to reuse a component from a system to another system of the same domain, i.e., the percentage of the code, which needs to be changed before reusing the component in a new system of the same domain. This latter aspect was adopted as the empirical reusability measure for our study. On the other hand, the internal characteristics we measure are divided into three categories: inheritance, coupling, and, complexity and size. The inheritance metrics we have used are 6 of the previously presented inheritance metrics. In the case of coupling metrics, we used at the design level, the 18 ones defined in table 2. Finally, the complexity and size metrics we have used are consigned in table 3.

Table 3. Complexity/size metrics.

Symbol	Definition
WMC	Complexity of a class defined as the sum of the complexities of the methods of this class.
RFC	Size of the Response Set of a class, defined as the set of methods in the class together with the set of methods called by the class's methods.
NOM	Number of local methods in a class.
CIS	Number of public methods in a class.
NPM	Average of the number of parameters per method in a class.
NOT	Number of trivial methods, which is marked as inline in C++, in a class.
NOP	Number of methods that can exhibit polymorphic behavior, e.g., virtual methods in C++.
NOD	Number of attributes in a class.
NAD	Number of user defined objects, i.e. abstract data types, used as attributes in a class.
NRA	Number of pointers and references used as attributes in a class.
NPA	Number of attributes that are declared as public in a class.
NOO	Number of overloaded operator methods defined in a class.
CSB	Size of the objects in bytes that will be created from a class declaration.

4 The empirical verification process

For leading our empirical verifications, we follow a four-step process: (i) hypothesis proposal, (ii) OO design metrics selection, (iii) predictive models generation, and (iv) models evaluation.

Machine-learning is of a significant help for the building of software quality predictive models. It is an important and prolific sub-field of Artificial Intelligence; it proposes many approaches like induction, deduction, analogy, probabilistic, soft-computing, etc. In order to study the pre-cited hypotheses, we have selected several algorithms belonging to various ML approaches and we have run them on software data collected from C++ and Java medium size applications. Some of the used ML algorithms are implemented in WEKA, an open source data-mining environment [24]. We have selected them in respect to three approaches. The first one is the induction of decision trees approach; it is represented by J48, an implementation of the well-known C4.5 algorithm [25]. It is a supervised learning algorithm that induces a classification model represented by a decision tree (or equivalent rules). The second approach is rules induction; Jrip, PART, and CN2 represent this approach. The former implements a propositional rule learner. It grows rules by greedily adding conditions with highest information gain, and after that, proceeds to incrementally prune each rule. On the other hand, PART allows the induction of rules by the iterative generation of partial decision trees; its main idea is to build a partial decision tree instead of an entirely explored one. Finally, CN2 [26] implements the covering technique, which represents classification knowledge as a disjunctive logical expression defining each class. The last approach is a hybrid one; it is illustrated by NBTree (Naïve-Bayes decision-Tree) that combines naive Bayesian classifiers and classifiers based on decision trees. It exploits a tree structure to divide the instances space into subspaces and to generate a naive Bayesian classifier for each subspace.

The computation of models accuracy is done thanks to a cross-validation procedure. It is helpful when the amount of data for training and testing is limited, which is our case; we try a fixed number of approximately equal partitions of the data, and each in turn is used for testing while the remainder is used for training. At the end, every instance has been used exactly once for testing. Table 4 resumes the obtained accuracies.

Table 4. Computed accuracies.

	Induction of decision trees: J48 or C4.5	Induction of rules: Jrip, PART, or, CN2	Induction of Bayesian decision trees: NBTree
Change impact – design coupling measures	73.85%	68.27%	66.75%
Fault-proneness – design coupling measures	77.50%	80.00%	70.00%
Fault-proneness - cohesion measures	73.70%	72.20%	65.00%
Fault-proneness – inheritance measures	73.80%	87.50%	72.50%
Reusability – design coupling measures	88.10%	63.10%	53.57%
Reusability - inheritance measures	73.80%	67.30%	46.50%
Reusability - complexity and size measures	89.30%	60.00%	52.40%

In terms of accuracy, and depending on which hypothesis and which ML algorithm we consider, we obtain pretty high results. They can be improved with an attribute pre-selection; we have noticed that some results are better than without the attribute selection. So, it will be interesting to repeat all the experiments with a prior attribute selection. On the other hand, a retro-propagation artificial neural network shows very interesting results (around 80% for the fault-proneness hypothesis). By analyzing table 4, one can conclude that for the systems we have studied, the design coupling between a class and its environment, the class position in the class hierarchy, and, at a lesser extent, the class cohesion, affect the class fault-proneness, as we defined it. On the other hand, decision trees-based algorithms show that design coupling metrics seem to be relevant also to assess reusability of OO medium size applications. It is also the case for the complexity/size of a class; it could be relevant to predict its reusability, for similar applications. They give also such good results to demonstrate the link between design coupling metrics and change impact.

In terms of selected internal metrics, import coupling seems to influence much more change impact than other types of coupling, since in most cases, the impact is mainly related to this type of coupling. On the other hand, it turns out that for the classes for which the number of static methods invocations, as well as the number of classes used with the target class, is large, import coupling (measured by the AMMIC metric) determines change impact. NBTree results added more details, and showed that coupling measured by CBONA and CBOU metrics also influences the impact. On the other hand, the classic CBO metric, and methods invocation, e.g., RFC_α (the number of methods that can potentially be executed in response to a message received by an object of that class), are identified as relevant design coupling measures for fault-proneness assessment. In the case of inheritance measures, NMI (the number of methods inherited) and DIT (the maximum length from the class to a root) are the two that are stated relevant in our study on fault-proneness. Finally, $LCOM_2$, $LCOM_5$, and $LCOM_1$ are the definitions of cohesion identified as relevant for predicting fault-proneness. Finally, the ML algorithms have revealed that export coupling (measured specially by OMMEC and OCAEC) has an impact on reusability. The complexity/size metric NPM (number of parameters per method) is also identified as a good indicator of reusability. In fact, combined with others like, NOP (number of polymorphic methods) and CSB (Class Size in Bytes), they allow us to determine if a software component is reusable.

These results confirm previous ones obtained on similar quality factors. Of course, more experiments on more data extracted from various and representative systems, are needed to confirm such conclusions. However, ML algorithms presented in this study are undeniable alternatives for software quality attributes prediction.

4 Conclusion

The main objective of this work is to improve the quality of software products, by learning from past projects and capturing some knowledge that will guide future developments. In this work, we have tried to show that machine learning is a feasible approach for software product quality prediction. The performances we have obtained vary depending on the targeted quality factor. The first important conclusion is relative to the accuracies obtained by the machine-learned predictive models. They are comparable for those obtained with other techniques, and even sometimes, higher! Further more, we know that for some ML algorithms, the results will be much higher with non-continuous numeric data. Some preliminary works on a pre-processing process tending (i) to translate continue attributes into discrete ones, and (ii) to select the more relevant attributes, based on statistical considerations, are currently done. They give promising results.

The main strength of such ML produced models is that we can incorporate them in an decision-making process, where, an expert system architecture keeps a good separation between what we consider as an expert knowledge (the produced models, e.g., rules, trees …) and the procedures that exploit this knowledge. However, they have to be confirmed and generalized by more experiments on more software data, extracted from various and representative applications. This is the challenge to produce relevant and reusable models.

References

1. L. Briand, P. Devanbu, W. Melo: An Investigation into Coupling Measures for C++. Proceedings of ICSE ‘97, Boston, USA, 1997.
2. S.R. Chidamber, C.F. Kemerer: A Metrics Suite for Object-Oriented Design. IEEE Transactions on Software Engineering, 20 (6), 1994, 476-493.
3. W. Li, S. Henry: Object-Oriented Metrics that Predict Maintainability. J. Systems and Software, 23 (2), 1993, 111-122.
4. L. C. Briand, S. J. Carrière, R. Kazman, J. Wüst: A Comprehensive Framework for Architecture Evaluation. International Software Engineering Research Network Report ISERN-98-28.

5. H. Lounis, H.A. Sahraoui, W.L. Melo: Defining, Measuring and Using Coupling metrics in Object-Oriented Environment. In SIGPLAN OOPSLA'97 Workshop on Object-Oriented Product Metrics, Atlanta, USA, 1997.
6. L.C. Briand, J. Wust, H. Lounis: Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs. In Empirical Software Engineering, an International Journal, March 2001, Kluwer Academic Publishers, 6(1):11-58.
7. Y. Mao, H.A. Sahraoui, H. Lounis: Reusability Hypothesis Verification Using Machine Learning Techniques: A Case Study. In Proc. of the 13th IEEE International Automated Software Engineering Conference, Hawai, October 13-16, 1998, 84-93.
8. J. Han: Supporting Impact Analysis and Change Propagation in Software Engineering Environments. In Proceedings of the STEP'97, London, England, July 1997, 172-182.
9. G. Antoniol, G. Canfora, A. De Lucia: Estimating the size of changes for evolving object-oriented systems: a Case Study. In Proceedings of the 6th International Software Metrics Symposium, Boca Raton, Florida, Nov 1999, 250-258.
10. D. C. Kung, J. Gao, P. Hsia, J. Lin, Y. Toyoshima: Class firewall, test order, and regression testing of object-oriented programs. In Journal of Object-Oriented Programming, Vol. 8, No. 2, May 1995, 51-65.
11. L. Li, A. J. Offutt: Algorithmic Analysis of the Impact of Changes to Object-Oriented Software. In proceedings of ICSM'96, 1996, 171-184.
12. L. C. Briand, J. Wüst, H. Lounis: Using Coupling Measurement for Impact Analysis in Object-Oriented Systems. In proceedings of the International Conference on Software Maintenance ICSM'99, Oxford, England, August 30 – September 3, 1999.
13. M. A. Chaumun: Change Impact Analysis in Object-Oriented Systems: Conceptual Model and Application to C++. Master's thesis, Université de Montréal, Canada, November 1998.
14. H. Kabaili: Changeabilité des logiciels orientés objet propriétés architecturales et indicateurs de qualité. PhD thesis, Université de Montréal, Canada, Janvier, 2002.
15. T.M. Khoshgoftaar, J.C. Munson: Predicting Software Development Errors Using Software Complexity Metrics. In IEEE journal on selected Areas in Communications, vol.8, n.2, February 1990.
16. A. Porter, R. Selby: Empirically guided software development using metric-based classification trees. In IEEE Software, March 1990, 7(2):46-54.
17. V. Basili, Condon, K. El Emam, R. B. Hendrick, W. L. Melo: Characterizing and Modeling the Cost of Rework in a Library of Reusable Software Components. In Proc. of the IEEE 19th Int'l. Conf. on S/W Eng., Boston, 1997.
18. M. Jorgensen: Experience with the Accuracy of Software Maintenance Task Effort Prediction Models. In IEEE TSE, August 1995, 21(8):674-681.
19. M.A. De Almeida, H. Lounis, W. Melo: An Investigation on the Use of ML Models for Estimating Software Correctability. In the Int. Journal of Software Engineering and Knowledge Engineering, October 1999.
20. H.A. Sahraoui, M. Boukadoum, H. Lounis: Building Quality Estimation models with Fuzzy Threshold Values. In "L'objet", volume 7, number 4, 2001.
21. R. Schauer, R. K. Keller, B. Laguë, G. Knapen, S. Robitaille, G. Saint-Denis: The SPOOL Design Repository: Architecture, Schema, and Mechanisms. In Hakan Erdogmus and Oryal Tanir editors, Advances in Software Engineering. Topics in Evolution, Comprehension, and Evaluation. Springer-Verlag, 2001.
22. M.K Abdi, H. Lounis, H. Sahraoui: Analyzing Change Impact in Object-Oriented Systems. In proceedings of the 32nd EUROMICRO Software Engineering and Advanced Applications Conference, Cavtat/Dubrovnik (Croatia), August 29-September 1, 2006.
23. H. Yazid, H. Lounis: Exploring an Open Source Data Mining Environment for Software Product Quality Decision Making. In proceedings of the Joint Conference on Knowledge-Based Software Engineering 2006 (JCKBSE'06), Tallinn, Estonia, August 28-31, 2006.

24. I.H. Witten, E. Frank: Data Mining: Practical Machine Learning Tools and Techniques with Java Implementation. Morgan Kaufmann Publishers, San Francisco, California, 2000.
25. J.R. Quinlan: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, 1993.
26. P. Clark, T. Niblett: The CN2 induction algorithm. In ML Journal, 1989, 3(4):261-283.

An Evaluation of the Design of Use Case Points (UCP)

Joost Ouwerkerk[1] and Alain Abran[1]

[1] École de Technologie Supérieure - ETS
1100 Notre-Dame Ouest, Montréal, Canada   H3C 1K3
joosto@gmaillcom; alain.abran@etsmtl.ca

Abstract. The increasing popularity of use-case driven development methodologies has generated an industrial interest in software size and effort estimation based on Use Case Points (UCP). This paper presents an evaluation of the design of the UCP measurement method. The analysis looks into the concepts, as well as the explicit and implicit principles of the method, the correspondence between its measurements and empirical reality and the consistency of its system of points and weights.

Keywords: Use Cases, Use Case Points, UCP, Function Points, FP, Functional Size Measurement.

1 Introduction

In the early days of measurement, a software project's size was measured by lines of code (LOC), and, since the early '80s, by functional sizing methods like Function Points Analysis (FPA) [ALB79, ISO 20926] [ABR03, ISO 19761]. With the subsequent increase in popularity of development methodologies based on use cases (UML, Agile, RUP) has come a concomitant interest in new methods for measuring the size of use-case-driven software projects.

The use case points (UCP) sizing method was initially designed by Gustav Karner of the company Objectory AB in 1993 [KAR93], and there has been little modification of it since then. UCP is an adaptation of FPA for measuring the size of projects, the functional requirements specifications of which are described by a use-case model. The use of UCP has been reported in [CAR05] [CLE06] [MOH05], and some commercial tools to help with the calculation of UCP have already emerged in the marketplace[13].

To evaluate the usefulness and validity of UCP sizing, this article proposes an analysis of the design of the UCP measurement method, looking into the explicit and implicit principles of the UCP method, the correspondence between its measurements and empirical reality, and the consistency of its system of points and weights. This evaluation approach is based on an evaluation process previously used for

[13] For example: Duvessa Estimate Easy Use Case - www.duvessa.com

investigating the design of cyclomatic complexity [ABR04], Halstead's metrics [ALQ05] and IFPUG function points [ABR94].

The structure of the article is as follows: section 2 introduces the UCP method, its origins and its procedures; section 3 presents the evaluation model; and section 4 the analysis results for the UCP method. In section 5, the results are discussed and the conclusions summarized.

2 Use Case Points

2.1 Origins

The use-case approach was developed by Ivar Jacobson while at Ericsson in the sixties. The idea was first described in 1987 [JAC87]. Like many of Jacobson's ideas, such as the sequence diagram, the concept of use cases was integrated into the Rational Unified Process (RUP) when Objectory AB was bought by Rational (now IBM) in 1995.

A use case is a simple but flexible technique for capturing the requirements of a software system. Each use case is composed of a number of *scenarios*, written in an informal manner in the language of the business domain to describe the interactions between the actors and a system.

In the Objectory and RUP process models, use cases are key to ensuring the traceability of requirements throughout the development cycle. It should be possible to trace a use case from requirements specification through to the architecture, design, code and testing phases?

In 1993, Gustav Karner [KRA93] proposed the UCP measurement method, an adaptation of FPA, to measure the size of software developed with the Objectory use-case approach. This method is aimed at measuring software functional size as early as possible in the development cycle.

2.2 Description

The design of UCP takes three aspects of a software project into account: 1) use cases; 2) technical qualities; and 3) development resources.

A) Unadjusted Use Case Points – UUCP

Use Cases are represented by a number of Unadjusted Use Case Points (UUCP). To derive the UUCP, the complexity of the system's actors and use cases must be evaluated. Table 1 summarizes the calculation of UCCP points:

- Each simple actor has a weight of 1,
- Each actor of average complexity has a weight of 2,
- and so on.

B) Technical qualities

Technical qualities are represented by a Technical Complexity Factor (TCF), which consists of 13 technical qualities (Table 3), each with a specific weight, combined into a single factor. To calculate the TCF, an expert must assess the relevance to the project of each technical quality, evaluated on a scale from 0 to 5 (where 0 is 'not applicable' and 5 is 'essential'). The weights are balanced in such a way that a relevance factor of 3 for each quality will produce a TCF equal to 1.

The TCF is thus the sum of all the relevance factors (one for each technical quality) multiplied by their corresponding weights plus two constants, C2 (0.1) and C1 (0.6): Table 3 lists the quality factors and their corresponding weights. Karner bases his design for these weights on the constants and weights of the FPA Value Adjustment Factors [ALB79].

C) Development Resources

Development resources are represented by Environment Factors (EF), also referred to as experience factors [CAR05]. The UCP model identifies eight such factors (Table 4) contributing to the effectiveness of the development team. To calculate the EF, an expert must assess the importance of each factor and classify it on a scale from 0 to 5 (0 meaning 'very weak'; 5 meaning 'very strong'). The selection of the weights is balanced such that a value of 3 for each factor will produce an EF of 1. The EF is the sum of all the factors multiplied by their weights and two constants, C2 (-0.03) and C1 (1.4).

Table 1: ACTOR Weights

Complexity	Definition.	Weight
Simple	System interaction via API.	**1**
Average	Average interaction system via protocol, or Human interaction via a command line.	**2**
Complex	Complex human interaction via a graphical user interface	**3**

Table 2: USE CASE Weights

Complexity	Definition	Weight
Simple	3 transactions or fewer; 5 analysis classes or fewer	**5**
Average	4 to 7 transactions; 5 to 10 analysis classes	**10**
Complex	Over 7 transactions; Over 10 analysis classes	**15**

Table 3: Technical Quality Factors – TCF

Factor	Description	Weight
F1	Distributed system	**2**
F2	Performance (response time or flow)	**1**
F3	Efficiency of user interface	**1**
F4	Processing complexity	**1**
F5	Reusability	**1**
F6	Installability	**0.5**
F7	Operability	**0.5**
F8	Portability	**2**
F9	Maintenability	**1**
F10	Simultaneous access	**1**
F11	Security	**1**
F12	Direct access for third parties	**1**
F13	Training features or online help	**1**

Table 4: Environmental Factors – EF

Factor	Description.	Weight
F1	Familiarity with the methodology	**1.5**
F2	Part-time status	**-1**
F3	Analysis capability	**0.5**
F4	Experience with the application	**0.5**
F5	Experience with object-oriented methodology	**1**
F6	Motivation	**1**
F7	Difficulty of the programming language	**-1**
F8	Stability of the specifications	**2**

The number of UCP is the product of these three components, UUCP, TCTP and EF.

For estimation purposes, the number of UCP can then be combined with a productivity constant (referred to as Mean Resources – MR) representing the ratio of man-hours per UCP. Karner proposes that each organization require its own productivity constant. The MR constant for the Objectory projects described in Karner's paper was approximately 20.

2.3 Related work on industrial applications of UCP

Four studies have been identified on the use of variants of UCP, and these are discussed next. No other detailed study has been identified documenting the use of UCP as is for estimation purposes.

On the basis of a single case study, [NAG01] reports that the UCP method can be more reliable than FPA to predict testing effort. The approach described in [NAG01] is a variant of Karner's, proposing nine technical qualities (instead of thirteen) associated with testing activities (for example, test tools and test environment) and ignores the development resource factors (EF). For the single project studied, the effort estimated by the UCP method was reported to be only 6% lower than the actual effort. The author himself stresses that this "*estimation technique is not claimed to be rigorous*". Of course, this study represents a single case study and lacks generalization power.

The adapted UCP method described in [MOH05] suggests that iterative projects need special rules to account for the ongoing reestimation of changing requirements during the course of a project. Realizing that an organization's resources are more or less constant, the adapted method replaces the resource factor (EF) by a simple increase in the productivity constant (MR).

In this variant of the UCP method, another dimension is introduced: the overhead factor (OF), which represents the effort of project management and another activities not directly related to functionality. For the two projects presented in [MOH05], the efforts estimated by UCP were 21% and 17% lower than the actual effort. Again, this study refers to only two projects and lacks generalization power.

The method used in [CAR05] is, in essence, the same as Karner's, but with the addition of a new "risk coefficient", specific to each project, to account for risk factors not already covered by the basic model (for instance, "reuse effort". The risk coefficient is a simple percentage of effort added according to the opinion of an estimation expert. The method described in the [CAR05] study was reportedly used with over 200 projects over a 5-year period, and produced an average deviation of less than 9% between estimated effort (using the UCP method) and recorded effort; no information is provided about the dispersion across this average, nor about the statistical technique used to build the estimation model or the classical quality criteria of estimation models such as the coefficient of determination (R^2) and Mean Relative Error. In brief, no documented details are given to support the claim about the accuracy of the estimates.

In summary, the UCP measurement method itself was modified in the three studies surveyed, and care must be exercised when comparing data from these three method variants, since no information about convertibility back to the original UCP is provided. Similarly, the empirical information given in these three studies cannot be generalized to a larger context due to either the very small sample (one or two case studies) or lack of supporting evidence.

3 Evaluation approach

As shown in [ALQ05], a measurement method can be analyzed by reverse engineering its design. Accordingly, the framework for measurement design proposed in [HAB06] is used here to study the measurement design of UCP, the terminology in [HAB06] being mostly based on ISO standards.

The analysis of the UCP measure that follows is structured according to the design activities identified in [HAB06]:

- Definition of measurement principles: knowledge and understanding of the concepts being measured, i.e. the model of the domain and the entities and attributes being measured;
- Definition of the measurement method: how the measurement makes the link between the empirical world and the numerical world;
- Definition of the operational procedures of measurement: the operational practices used to apply the method.

[HAB06] reminds us that there are different types of mathematical scales, and categorical rules for transforming one scale to another. The scale types typically used are: nominal (values are non-ordered categories), ordinal (values are ordered categories), interval (there is a concept of relative “distance” between the values) and ratio (absolute intervals with a meaningful zero-value).

4 Analysis of the UCP measurement design

4.1 Empirical principles

UCP is aimed at measuring the functional size of a software system described by a functional requirement specification written in use-case form.

Although based on the FP structure, [KAR93] proposes several other factors, including non-functional requirements as well as development resource characteristics.

Table 5 lists the five entities currently being measured by the UCP method described in [KAR93].

As identified in [SMI99], the fairly vague definition of a use case poses an important problem for the calculation of UCP. The way in which a use case is written is not standardized; it can be identified and documented at any level of granularity. What is an elementary use case? How can we even characterize the level of granularity of a particular use case? Are use cases being characterized in a consistent manner by one or more analysts working on the same project, and across projects?

Successful measurement using the UCP method will therefore directly and strongly depend on a degree of uniformity in the writing of use cases from one project to another.

For the entities listed in Table 5, eleven different attributes have been identified (Table 6), each of which is taken into account and measured by the UCP method described in [KAR93].

The set of entities and attributes measured by UCP is illustrated in Figure 1. The UCP measurement process clearly includes a mix of several entities (actors, use cases, requirements, team, programming language) for which distinct attributes are quantified and then combined.

In summary:

- The resulting units of measurement are unknown and unspecified despite the UCP label.
- It is obvious from Tables 5 and 6 that the end-result cannot be uniquely expressed in terms of use cases.

Table 5: ENTITIES

Entity	Description
Actor	A use case, as defined by [JAC87], describes the interaction between the actors and the system. The actor is any agent (machine or human) that acts upon system functionality.
Use Case	A simple functional requirement description for a specific goal, written in the form of a sequence of interactions between an actor and the system.
Specification of requirements	The set of planned requirements for a system, including the functional requirements (written in use-case form) and other non-functional requirements.
Development team	The human resources participating in the project of designing, programming and testing the system.
Programming language	The computer programming language used by the development team to code the software system (Java or C++, for instance).

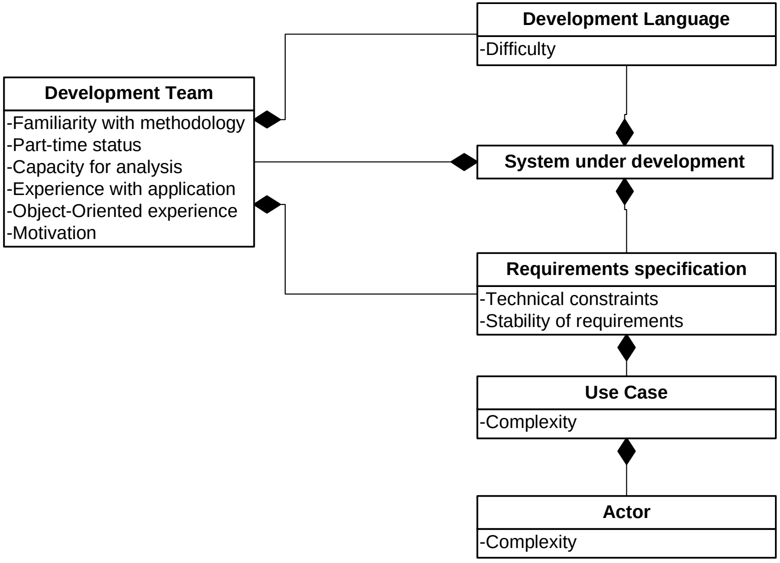


Figure 1: Set of entities and attributes measured in UCP

Table 6: ATTRIBUTES

Entity	Attribute	Measurement rule
Actor	Complexity (of actor)	The type of complexity (simple, average or complex) of the interaction between the actor and the system.
Use case	Complexity (of use case)	The type of complexity (simple, average or complex) measured in the number of transactions.
Specification of requirements	Relevance of the technical quality requirements	The level of relevance (from 0 to 5) of each of the 13 known non-functional qualities
	Stability of requirements	The level of stability (from 0 to 5) of the functional and non-functional requirements
Development team	Familiarity with the methodology	The level (from 0 to 5) of skills and knowledge of the development methodology in use for the project.
	Part-time status	The level (from 0 to 5) of part-time staff on the team
	Analysis capability	The level (from 0 to 5) of analysis capabilities of the development team with respect to project needs.
	Application experience	The level (from 0 to 5) of team experience with the application domain of the system
	Object-oriented experience	The level (from 0 to 5) of team experience with object-oriented design
	Motivation.	The level (from 0 to 5) of team motivation
Programming language	Difficulty	The level (from 0 to 5) of programming difficulty

4.2 Scale types of the attributes

The UCP measurement method includes two categories of attributes for the six entity types being measured: types and levels. These attributes are quantified using a variety of scales types, and then they are combined. This section analyzes these scale-type manipulations and identifies the corresponding measurement issues.

The “complexity” attribute, assigned to actors and use cases, is at the core of UCP (collectively defined as the UUCP factor). It is categorized as being of the ordinal scale type using a scale of three values: simple, average and complex. Thus an actor categorized by the measurer as “simple” is considered less complex than an “average” actor, and an “average” actor less complex than a “complex” actor. The scale is similar for use cases: the same category labels are used (simple, average and complex), however it cannot be assumed that the categories and the categorization process are similar, since different entity types are involved.

The technical and resource factors are also all evaluated through a categorization process on an ordinal scale, but one with integers from 0 to 5 (0, 1, 2, 3, 4 or 5); it must be noted that these numbers do not represent numerical values on a ratio scale, but merely a category on an ordinal scale; that is, they are merely ordered labels and not numbers. Thus, a programming language assigned a difficulty level of 1 is considered to be less difficult for the development team than a programming language of difficulty level 2, but cannot be considered to be exactly one unit less difficult than one categorized as having a difficulty level of 2 because these levels are being measured on an ordinal scale. There is indeed no justification provided in the description of the UCP model to support a ratio scale (for example, that a programming language of factor 4 is twice as difficult as a programming language of factor 2). The levels must therefore be regarded as being on an interval scale. It must also be noted that, even though they have the same labels, e.g. 1,2,3, etc., the intervals are not necessarily regular; for example, each label might represent a different interval, and each interval may not be, and does not need to be, regular within an attribute being measured – see, for instance, the measurement rules for the Technical Adjustment Factors in FPA [ABR94].

4.3 The measurement method for the entities

The measurement process for assigning a value to the complexity attributes of actors and use cases is based on categorical rules. For example, if an actor acted on the system by means of a graphical user interface, then the value assigned to it is labeled “complex”; as explained in the previous section, this value is on an ordinal scale.

For use-case entities, the measurement process for assigning a value comes first from counting the number of transactions or the number of analysis classes, and then, as a second step, looking up the correspondence rules in a two-dimensional table (transactions and analysis classes) to assign a corresponding ordered label of an ordinal-type scale. Thus, if a use case contains four transactions, it is assigned the category label “simple”.

Both the technical factors and the resources are evaluated qualitatively by a measurer. The UCP method does not prescribe any rule for assigning a value to, for example, the level of familiarity with a methodology. Without criteria for the evaluation of levels, the UCP measurement process lacks strict repeatability, or even reproducibility; one measurer (beginner or expert) can easily evaluate a factor differently from another measurer.

4.4 Correspondence mapping between the empirical and numerical worlds

The principle of homomorphism requires that the integrity of attribute ratios and relationships be maintained when translating from the empirical model to the numerical model.

Below, we evaluate these numerical correspondence mappings in UCP for each attribute of the empirical model.

Actor complexity: If we accept that an application programming interface (API) represents less functionality than a command-line interface, which in turn represents less functionality than a graphical user interface, then we can say that the categorization of the actors by their type of interface represents a valid correspondence. But if, after assigning a "weight" of 1, 2 or 3 to the actor types, the model uses these ordered values in sums and products, then it is effectively making a translation from an ordinal scale (complexity categories) to a ratio scale (UCP weights) without justification, which is not a mathematically valid operation. Furthermore, there is no documented data to demonstrate that a graphical interface represents three times more "functionality" than an application programming interface.

Use-case complexity: The UCP model transforms the measurements of use-case complexity from a ratio-type scale (the number of transactions or classes of analysis) into an ordinal-type scale (complexity categories), and then back to a ratio-type scale (UCP weights). The arbitrary assignment of the weights (5 for simple, 10 for average and 15 for complex) could have been avoided if the number of transactions or classes of analysis had been kept as numbers on a ratio-type scale (rather than losing this quantitative precision by mapping them to only three ordered categories with arbitrarily assigned values of 5, 10 and 15.

Technical qualities: the UCP model does not propose any criteria for measuring the attribute values of technical qualities. The result is an entirely subjective measure. To address the same kind of arbitrary assignment of values in the "technical adjustment factors" of the FP model, the IFPUG organization worked out detailed evaluation criteria for each of the non functional characteristics of the system and documented these in its Counting Practices Manual.

Both the constants and weights of UCP's technical qualify factors (TCF) calculation are derived directly from the Albrecht's FP model. It has been noted in [ABR94] that the TCF suffer from a large number of inadmissible mathematical operations moving up and down the scale types, and then using these numbers in inappropriate ways.

Resource factors: the UCP model does not propose any criteria for measuring resource factors. Once again, this "measurement" is effectively a subjective interpretation by a measurer.

It should be noted that Karner [KAR93] had indicated that the EF constants and weights were preliminary and estimated. That being said, more recent sources like [CLE06] and [CAR05] have not revisited this issue and have integrated the same values without modification into their own measurement models of UCP variants.

4.4 Inadmissible numerical operations

The formula to calculate the number of UCP is as follows:

$$\mathbf{UCP = UUCP * TCF * EF}$$

Albrecht [ALB79] had initially characterized FP as being “dimensionless coefficients”. As UCP is an adaptation of FP for use cases, some have taken for granted that the UCP are numbers without units: that is, it is assumed that one arrives at the UCP number by multiplying (apparently) dimensionless constants with (again, apparently) dimensionless weights with values on ordinal or nominal scales. This issue has been documented in [ABR94] and [ABR96].

More specifically, for UCP, the final UUCP value is calculated by multiplying the number of use cases and actors of each type of complexity by their weights. In doing so, the ordinal-scale values (categories of complexity) are transformed into interval-scale values. These are than multiplied, resulting in a value on a ratio scale, another algebraically inadmissible mathematical operation.

This analysis applies equally to TCF and EF factor calculations, which, in addition to the transformations of ordinal-type scale into interval-type scale (confounding numerical values with the ordering of categories by numerical label), also introduce multiplications with ratio-scale constants. [ABR94] described this same error in the FP measurement method, the origin of the EF and TCF calculations.

It has further been pointed out, in [ABR94, ABRA96], that the interaction between technical factors would have been better accounted for by multiplying the factors together rather than adding them. This analysis also applies for the EF factors in this case. In summary, TCF has inherited most of the defects of FP – see Appendix, and in some cases has compounded them by adding additional inadmissible mathematical operations.

5 Observations and conclusions

To summarize, this investigation of the measurement principles behind the UCP method has raised the following measurement issues:

- Use cases can be described at various levels of granularity – the UCP method does not take this into account and does not describe any means to ensure the consistency of granularity from one use case to another. The impact is the following: quantities as the outcome of UCP are not necessarily comparable and a poor basis for benchmarking and estimation.
- The UCP measurement method measures several entities (actors, use cases, requirements, etc.) and attributes (complexity, difficulty, etc.). Combining, as it does,

too many concepts at once makes it a challenge to figure out what the end-result is from a measurement perspective. The impact of this is that the end-result is of an unknown and unspecified entity type; that is, we do not know what has been measured.

• The UCP measurement method is based on the constants and weights of Albrecht's FP Value Adjustment Factors without supporting justification.

• The evaluation of attributes is performed by a measurer without criteria or a guide to interpretation – a 5 for one measurer could be a 2 for another. Therefore, repeatability as well as reproducibility could be very poor.

• UCP method calculations are based on several algebraically inadmissible scale type transformations. It is unfortunate that this has not yet been challenged, either by UCP users or the designers of subsequent UCP variants.

The measurement of functional size using use cases (therefore, very early in the development life cycle) represents an interesting opportunity for an industry which is increasingly using RUP and Agile use-case-oriented methodologies. Unfortunately, the UCP method suggested by Karner appears fundamentally defective: by adopting the basic structure of FP, UCP has – from a measurement perspective – inherited most of its structural defects.

There have been claims for the relevance of UCP for estimation purposes, but with very limited empirical support. Even when empirical support is claimed, it is based either on UCP variants and anecdotal support or on undocumented evidence.

The idea of measuring size and estimating effort with use cases remains attractive, and Karner's method represents a first step towards that objective. That said, the lessons learned through this investigation of the UCP design clearly indicate that:

- On the one hand, significant improvements to UCP design are required to build a stronger foundation for valuable measurement;
- On the other hand, much more robust and documented empirical evidence is required to demonstrate the usefulness of UCP for estimation purposes.

Appendix: FPA use of scales in the measurement of Data Files (EIF and ILF)

Step	Entities	Operations	Scale: From	Scale: To	Math. Validity	Implicit transformation
Count elements of data and records	Data	Count	Absolute	Absolute	Yes	No
	Record	Count	Absolute	Absolute	Yes	No
Execute data algorithm	Data	Identify range	Absolute	Ordinal	Yes	Yes, and loss of information
	Record	Identify range	Absolute	Ordinal	Yes	Yes, and loss of information
	Function of ranges of (data,record)	Position in matrix	Ordinal	Nominal	Yes	Yes, and loss of information
	Function of position in matrix	Name and order	Nominal	Ordinal	No	Yes, and loss of information
	Function of perceived values	Assign weights	Ordinal	Ratio	No	Yes, and loss of information
Add all points	Weights of internal files	Add	Ratio	Ratio	Yes	No
	Weights of external files	Add	Ratio	Ratio	Yes	No
	Weights: internal + external	Add	Ratio	Ratio	Yes	No

References

[ALB79] Albrecht, A.J, “Measuring Application Development Productivity,” Proceedings of IBM Application Development Joint SHARE/GUIDE Symposium, Monterey, CA, USA, 1979, pp. 83-92.

[AND01] Anda, B., Dreiem, H., Sjøberg, D.I.K. and Jørgensen, M. “Estimating Software Development Effort Based on Use Cases – Experiences from Industry,” 4th International Conference on the Unified Modeling Language (UML2001), October 1-5, 2001, Toronto, Canada, Springer Verlag, pp. 487-502.

[ABR94] Abran, Alain; Robillard, P.N., “Function Points: A Study of their Measurement Processes and Scale transformation,” Journal of Systems and Software, vol 65, 1994, pp. 171-184.

[ABR96] Abran, Alain; Robillard, P.N., “Function Points Analysis: An Empirical Study of Its Measurement Processes,” IEEE Transactions on Software Engineering, Vol. 22, no. 12, December 1996, pp. 895-910,

[ABRA03] Abran, A., Desharnais, JM., Oligny, S., St-Pierre, D., Symons, C., COSMIC-FFP version 2.2 – The COSMIC Implementation Guide to ISO/IEC 19761, École de technologie supérieure, Université du Québec, Montréal, Canada, 2003 url: www.gelog.etsmtl.ca/cosmic-ffp

[ABR04] Abran, A., Lopez, M., Habra, N., “An Analysis of the McCabe Cyclomatic Complexity Number,” 14th International Symposium on Software Measurement IWSM-Metrikon, 2004, Magdeburg, Germany, pp. 391-405.

[ALQ05] Al Qutaish, R. E., Abran, A., “An Analysis of the Design and Definitions of Halstead’s Metrics,” 15th International Symposium on Software Measurement (IWSM), 2005, Montréal, Canada.

[CAR05] Carroll, Ed, “Software Estimating Based on Use Case Points,” The Cursor, Software Association of Oregon, Website: http://www.sao.org/newsletter

[CLE06] Clemmons, Roy K., “Project Estimation With Use Case Points,” Crosstalk, February 2006, pp. 18-22

[COC97] Cockburn, Alistair, “Structuring Use Cases with Goals,” Journal of Object-Oriented Programming, September-October, 1997, and November-December, 1997.

[HAB06] Habra, N., Abran, A., Sellami, A., “A Framework for Software Measurement Design,” submitted for publication.

[ISO19761] ISO/IEC19761:2003, Software Engineering – COSMIC-FFP – A Functional Size Measurement Method, International Organization for Standardisation – ISO, Geneva, 2003.

[ISO20926] ISO/IEC 20926: 2003, Software engineering – IFPUG 4.1 Unadjusted functional size measurement method – Counting Practices Manual, International Organization for Standardisation – ISO, Geneva, 2003.

[JAC87] Jacobson, Ivar, “Object Oriented Development in an Industrial Environment,” OOPSLA ’87 Proceedings, October 4-8, 1987, pp. 183-191.

[JAC97] Jacquet, Jean-Philippe; Abran, Alain, "From Software Metrics to Software Measurement Methods: A Process Model," Third International Symposium on Software Engineering Standards, ISESS '97, Walnut Creek, CA. June 2-6, 1997.

[KAR93] Karner, Gustav, "Resource Estimation for Objectory Projects," Objective Systems SF AB, 1993.

[MOH05] Mohagheghi, Parastoo, "Effort Estimation of Use Cases for Incremental Large-Scale Software Development," IEEE International Conference on Software Engineering – ICSE '05, May 15-21, 2005, St.Louis, MI, USA, pp. 303-311.

[NAG01] Nageswaran, Suresh, "Test Effort Estimation Using Use Case Points," Quality Week 2001, San Francisco, CA, USA, June 2001.

[RIB01] Ribu, K., "Estimating Object-Oriented Software Projects with Use Cases," 2001, University of Oslo, Oslo, Norway. Website: http://www.stud.ifi.uio.no/~kribu/oppgave.pdf

[SMI99] Smith, John, "The Estimation of Effort Based on Use Cases," Rational Whitepaper, Rational Software, 1999.

A Proposal for Analysis and Prediction for Software Projects using Collaborative Filtering, In-Process Measurements and a Benchmarks Database

Yoshiki Mitani[1,2], Nahomi Kikuchi[1], Tomoko Matsumura[2], Naoki Ohsugi[2], Akito Monden[2], Yoshiki Higo[3], Katsuro Inoue[3], Mike Barker[2], Ken-ichi Matsumoto[2]

[1]IPA/Software Engineering Center (SEC), Tokyo, JAPAN
{y-mitani|n-kiku}@ipa.go.jp
[2]Nara Institute of Science and Technology (NAIST), Nara, JAPAN
{ymitani|tomoko-m|naoki-o|akito-m|mbaker|matumoto}@is.naist.jp
[3]Osaka Univ., Osaka, JAPAN
{y-higo|inoue}@ist.osaka-u.ac.jp

Abstract. This paper proposes a new method for developing predictions and estimates for ongoing projects by comparing in-process measurements of the current project with benchmark data from previous projects. The method uses collaborative filtering to identify groups of similar projects in the benchmark database and then to develop predictions and estimates based on the in-process measurements of the current project and the comparison data from the similar projects. The authors base this proposal on experiments with multidimensional in-process project measurement in a middle-scale multi-vendor development that lacked transparency in its processes. The authors' measurement trial verified the usefulness of the measurement methods, especially in project management, as reported in the paper.

Keywords: Empirical Software Engineering, In-process project measurement, Collaborative filtering, Software project database.

1 Introduction

This paper first provides a bird's-eye view of past research by the authors and fundamental methods. Then the paper explores a new project measurement and feedback method which has evolved from that past research.

Specifically, the paper describes the function and structure of a project measurement platform called the Empirical Project Monitor (EPM). The Empirical Approach to Software Engineering (EASE) project, an academic based project for collaboration between industry and academia, developed EPM [1][2][3][4]. Next, it presents experimental results from the application of EPM and related tools in a governmental project for multi-vendor software development called the Advanced Software Development (ASD) project [5][6]. After that, it presents a project to collect benchmark data from software projects which has collected data from over 1000 projects in 15 software companies [7]. The Software Engineering Center (SEC), an industry-based organization for collaboration of industry and academia, conducted this project [8].

After that, the paper introduces a method for data analysis using collaborative filtering technology which is effective for data sets with missing elements [9][10]. The paper presents two trials of this method of data analysis.

Finally, the paper describes a general method for performing such analysis and projections using dynamic measurements of software process and a database of past project measurements, based on the described research experiences. We propose to experimentally verify this proposed method in future research.

2 EPM: The in-process project measurement platform

EPM automatically collects software development management data from development tools such as a configuration management system, Concurrent Versioning System (CVS), bug tracking system, GNATS, and mailing list management system, mailman. Drawing especially from the configuration management system, EPM automatically collects source code and operational histories of source code development, the basic information concerning transitions that occur in the software development process. Fig. 1 shows an example of its displays.

EPM translates collected data into a standard XML format and stores them in a relational database for analysis. The EPM analysis functions display information in visual formats. The information includes changes in the source lines of code, timing analysis of check-in and check-out, changes in bug numbers, analysis of inter-company mail volumes, and the Software Reliability Growth Model (SRGM) curve.

By using EPM with such basic development tools as a configuration management tool, bug tracking tool, and mailing list management tool, a software project can receive the benefits of automatic measurements and visually presented analyses of project data without the burden of intrusive manual tracking.

3 Experience with in-process project measurement

3.1 Target project description and measurement

The ASD project started in Spring 2005. Funded by the government, the project focuses on development of kernel software for an experimental information system platform for collecting probe information. This probe information system will collect car location information from various automotive elements called probe cars, such as taxis, trucks, and busses. From this information, the probe information system will generate various useful public information formats.

The project period is two years, separated into two parts. The project is currently in the integration test phase of the 2nd part.

The project was organized as a development consortium, composed of seven companies, including six major software development companies and an automobile manufacturer. The automobile manufacturer acts as the evaluator and the other six companies develop the platform. One of the six companies acts as Project Manager (PM). The six companies are rivals in the probe information system field, so the project clearly distinguishes between collaboration and competitive materials. Information in the collaborative field is shared and in the competitive field is confidential. For example, detail design, source code, and source line of code (SLOC) productivity are confidential. However, the PM needs SLOC information for meaningful project management. Normally this situation would force the PM into a kind of blind management. During the companies' individual development phase, management would be based on declarations. Only in the inter-company integration test phase would all members share the real situation of the developed software.

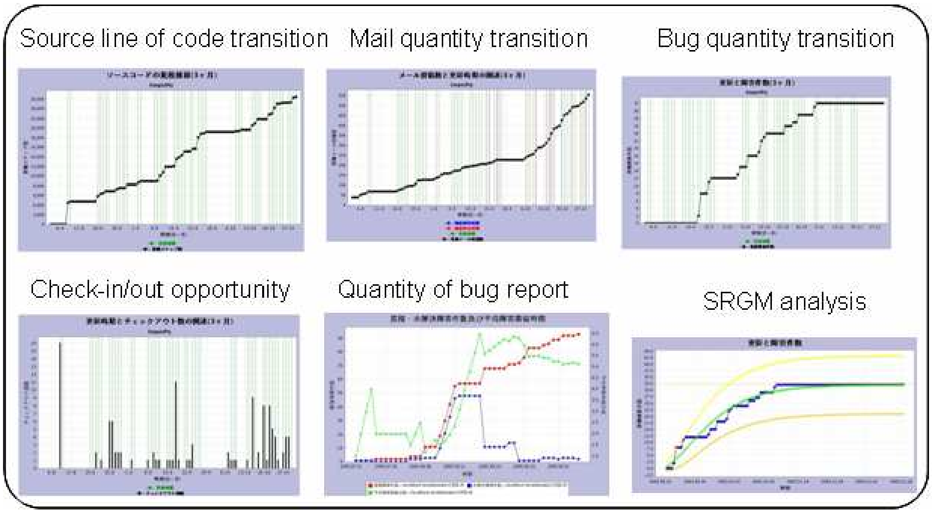


Fig.1 Empirical Project Monitor (EPM) display example

The target software is written in C/C++ and runs on several Linux servers with a relational database for data processing and personal computers for data display. Each consortium company measured project data, which was collected by SEC and analyzed for software engineering research. Analyzed data was fed back to the individual companies with respect for confidentiality. The PM was provided with a bird's-eye view of the total information, again with respect for confidentiality. This allowed more than the blind management that had been expected.

For this project, the following five methods of measurement, shown in Fig. 2, were used:

1) EPM measurement and analysis

EPM collected development process and product information, and produced analysis results.

2) Collection and analysis of review reports

An electronic data form with 30 items was used to collect information concerning basic and detailed design reviews.

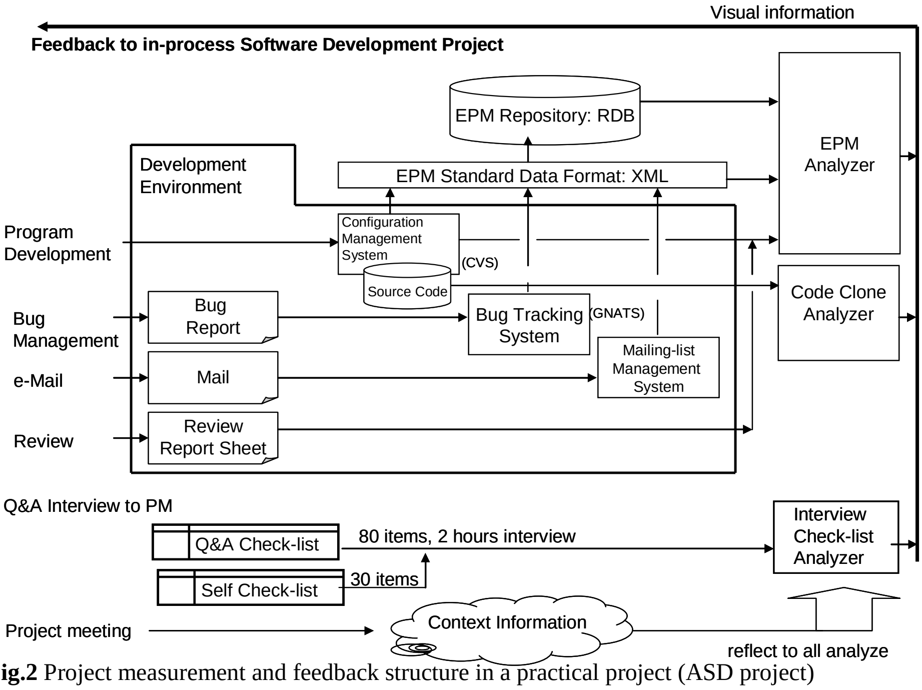


Fig.2 Project measurement and feedback structure in a practical project (ASD project)

3) Code Clone Analysis

A code clone is a code fragment in source code which is identical or similar to each other. Code clone fingerprints such as code clone distribution or content ratio represent software product characteristics. In this trial, we used CCFinder[11], which is a code clone detection tool. CCFinder displays the result of code clone detection by using Scatter Plot.

4) Questionnaire and interview of project leader and PM to collect project context information

SEC developed a questionnaire for collecting context information which includes 30 self-assessed items and 80 items for interviewing. These lists were developed in the context of the PMBOK knowledge areas. The self-assessment and interview collect various context data items which are difficult to obtain through automatic tools such as EPM.

5) Collect project context information by participation in project meetings

To collect more data about the project context, research staff attended all the project meetings. This was very useful in collecting information that could not be collected in other ways.

3.2 Results of the project measurement

In this project, the measurement effort brought development out of the black box into the daylight and helped to form a consensus about how to handle project management. For example, the following characteristics of this project were identified:

- Measurement provided a bird's-eye view of each company's project based on transitions in source line of code count and transitions in bug numbers.

- Code clone analysis helped identify the origin of source code and the development approach taken by different groups. For example, this analysis helped identify whether they mostly used code developed from scratch, by cut and try methods, or appropriated and reused code. The analysis also suggested characteristics about the source code such as whether it was produced by a less experienced coder or not, possible issues with future code maintenance, and the status of the refactoring process.

The project structure did not make source code from individual companies available to the project manager, so it was very useful to share and discuss code clone data between the development companies and the project manager. The fact that the project manager referred to code clone analysis data caused positive effects on the project. For example, one company explained their system design concept when a code clone was detected. This clone was the result of design considerations intended to increase future extendibility. This kind of discussion helped raise morale and provide opportunities for inter company coordination.

-Analysis of file renewal suggested differences in the development process, such as use of waterfall type process or cut-and-try development. This analysis clearly showed the stability of file renewal, the impact of design changes, and attention to bug detection in the late developed process.

- Analysis of bug reports showed clear relationships between various bug factors. In particular, analysis of relationships between the bug injection process and the bug detection process, along with consideration of when bugs should ideally be detected, were particularly useful in evaluating the early development process.

- The analysis of review reports clearly indicated the different attitudes towards the review process. Some companies invested significant effort in the review process, reducing problems in later stages of their waterfall development process. Other companies slighted the review process, expecting problems to be caught by later testing instead. We could expect some conflict on software parts from these different companies will be combined at the system integration test phase.

- The leader questionnaire and interview, recorded using checklists, provided information about the development structure as part of the context information which was generally confidential inside each company in the consortium. By providing some part of this information to the project manager, the project manager could understand how the different companies worked, allowing better project management.

For example, development in some companies with low code clone content was largely development from scratch, with some cut-and-try development in the logical processing portion of the project. Another company with high code clone content had a large level of reuse, and considered a key factor in success of this development to be adaptation of reused code. Finally, one company apparently expected the integration test to uncover problems instead of finding them by review activities.

4 Post-process benchmark data collection

4.1 Benchmark data collection from over 1000 projects and building a national database

The SEC has started to collect software project benchmark data from industry to build a national level database. As the first step in this process, the SEC has collected data from 1009 projects in 15 software industries. Preliminary analysis results have been published as the Software Data White Paper [12].

The data items collected were defined by the SEC in reference to data items previously collected by Japanese software industries and also data items collected by the International Software Benchmark Standard Group (ISBSG). The list contains about 490 items in 10 categories. An example is shown in Table 1. The data items are collected through an electronic data form. Preliminary analysis has identified some useful database attributes such as program size, total effort, productivity, reliability, and some correlations between basic data items.

Table 1 Example of Benchmark Data Items

Classification	Items
General items charactering projects	Type of development, new development, new customers, level of success
Application domain	Domains, types of application, characteristics of users
System characteristics	Usage of ERP, development platforms, development languages
Development procedure States of projects	Life cycle models, usage of tools, rate of reuse States of development teams, work environments
Requirements management	Ambiguity, commitment by users
Personal skills	Skills of Project Manager (PM), skills of team members
Development size	Function Point (FP) methods, FP, Source Line Of Code (SLOC)
Term	Terms of development (actual, planned)
Development effort	Total efforts (actual, planned), efforts by phases (actual, planed)
Quality	Total defects, defects by phases

4.2 Collaborative filtering of the benchmark database

The collected database includes many data sets with missing elements, and various kinds of projects. To analyze them required a technology that can handle missing elements and perform grouping and categorization of the projects. Collaborative filtering technology was applied to group similar projects from data sets with missing elements. A key feature of collaborative filtering for this application is that it can analyze data sets directly without any special operation for missing elements in included data sets or any special variable selection.

To validate the technology, a kind of experimental project prediction was performed using the project benchmark database. First, benchmark data from one project was selected as a key, and one data item from it, such as total development effort, was hidden. Next, the collaborative filtering tool retrieved a group of projects with similar data sets. Then the project total effort was estimated from the retrieved set of similar data and compared with the hidden real value. This makes a prediction or estimate based on the similar projects found in the SEC database. Ohsugi et al's research [13] (in Japanese) reports on this as a case study, but it suggests the potential of this prediction method. In this experiment, "total effort" was selected as an estimation target item from the 490 data items collected. The filtering key data consisted of 97 data items of planning data and actual data from the beginning to the end of the detailed design phase. Data from 378 projects which had no missing "total effort" measure were selected from the data on 1009 projects as target data for filtering. In the data for selected projects, the ratio of missing data items to total data items was 67%. Collaborative filtering allowed estimation of the total effort based on several other indicators. The average relative error, comparing the predicted value to the measured value, was 0.64. This result suggests the usefulness of this method despite the significant missing elements in the database. This research was only a case study, but indicates the importance of the activity of the SEC in creating a national database for software project benchmarks.

The ASD project has collected planning data and actual results from the beginning to the end of the basic design phase. The project total effort was estimated using the collected partial benchmark data as a key and the SEC benchmark database. After the end of the project, this can then be compared to final actual results data. In other words, the final results of the project will be compared with an estimate predicted using the current project benchmark data and the SEC database of previous software projects.

For example, in this experiment, the collaborative filtering tool calculated a project similarity grade indicating project similarity in 10 steps from 0.0 to 1.0, which was used to illustrate a similarity distribution graph. In the ASD project trial, collaborative filtering using the partial benchmark data from five companies at the end of the basic design phase retrieved about 70 similar projects in the similarity range from 0.9 to 1.0 from the 1009 projects in the SEC database. Both manual review and some statistical processing allow extraction of useful information for project operation from the characteristics of the retrieved group of similar projects.

Since the ASD project is now in the integration test phase, final evaluation results for this method have not yet been completed. However, we expect to verify the usefulness of this method in this project and confirm Ohsugi's research results.

5 A proposed method for using in-process measurements

Accumulated in-process data about process and product form a valuable database after completion of the project. However, it is not easy to use this information during the project with only simple accumulation of data.

Generally, in-process measurements are plotted with time as the horizontal axis and changes in various indices as the vertical axes. Typically the macro trend of the changes has meaning instead of the absolute numerical values. In most cases, the visual patterns of the graph or chart provide useful guidance for project management and operation.

As a main point of this paper, the authors propose in-process measurements and groups of similar projects extracted from the project benchmark database as described in 4.2. Fig. 3 illustrates the outline of this method, and the following describes the procedure. Number in parentheses corresponds to those in Fig. 3.

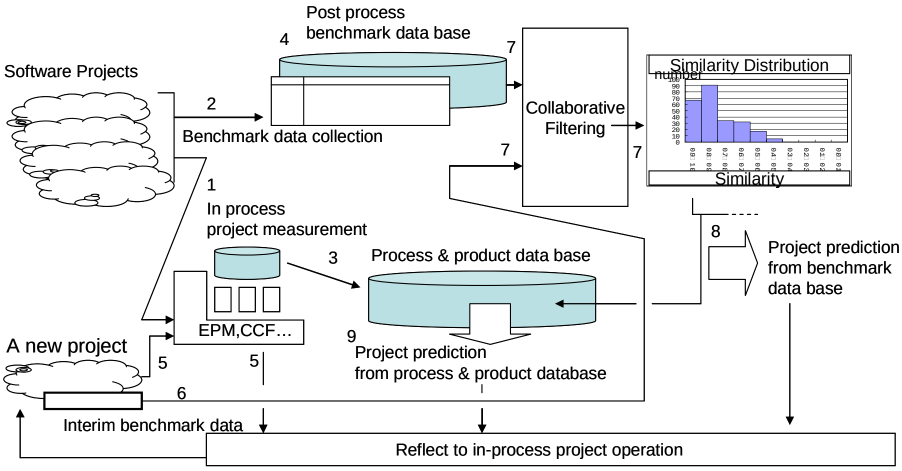


Fig.3 Project prediction by collaborative filtering with two kinds of project database

- First, from every project, benchmark data as described in 4.1 and measurement data about process and product as described in 3.1 are collected in a dataset (1)(2). This data is accumulated in a database (3)(4)
- Second, for a new project, interim benchmark data is collected, and collaborative filtering used to retrieve a group of similar projects from the benchmark database (7)
- Finally, the process and product measurements for the group of similar products (8) are used to generate estimates for the new project (9). These data, predictions based on the benchmark database (8) and in-process data measurements (5) are referenced to project operation in all.

6 Initiatives for future research

In early 2006, the authors initiated a new plan to experimentally validate the above approach. The plan includes the following elements.

1. First year: Development of easy-to-use distribution kit of measurement platform (EPM tools).
2. Second year: Execute practical experiment of measurement and database construction with ten trial projects.

The project measurement, analysis, and feedback mechanism shown in Fig. 3 depends on construction of a database of project measurements. Such databases built inside companies are useful, however, a national database like the SEC benchmark database is considered more valuable.

Popularizing the measurement platform is an important first step in process and product data measurement. The new plan includes distribution of a useful tool kit or environment for measurement and practical experiments applying it. The authors' aim is to make the measurement platform highly popular and to share the evaluation results from the practical experiment widely, then to build a national level mechanism which includes the measurement database shown in Fig. 3.

7 Conclusion

In the field of software project measurement, there are two broad kinds of measurement, post-process collection of benchmark data and in-process measurements of process and product.

In Japan, the SEC has been building a national database for benchmark data since 2005.The authors' experiment has demonstrated one method for using this database. The method provides predictions or estimates for projects by applying collaborative filtering to retrieve groups of similar projects from the benchmark database using interim measurements of the current project as the retrieval key.

In terms of in-process project measurements, useful tools such as the measurement platform called EPM and code clone analysis tool called CCFinder have been provided, and experimentally shown to be useful in projects such as the ASD project. This paper provided a bird's-eye view of the authors' work and proposed an approach to build and use a database of process and product measurements, integrating the previous experiments of the authors. The approach uses collaborative filtering to extract groups of similar projects from a benchmark database using interim benchmark data from a current project. Finally, the authors describe a future initiative to develop the proposed environment and verify its usefulness.

Acknowledgements

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] EASE project: http://www.empirical.jp/English/index.html

[2] Yoshiki Mitani, Mike Barker, Koji Torii, Seishiro Tsuruho: An Experimental Framework for Japanese Academic-Industry Collaboration in Empirical Software Engineering Research, International Symposium on Empirical Software Engineering□ ISESE)2004,Vol.2, Redondo Beach, USA, Aug. (2004) pp.35-36

[3] Masao Ohira, Reishi Yokomori, Makoto Sakai, Ken-ichi Matsumoto, Katsuro Inoue, Michael Barker, Koji Torii: Empirical Project Monitor: A System for Managing Software Development Projects in Real Time , International Symposium on Empirical Software Engineering□ISESE□2004, Vol.2, Redondo Beach, USA, Aug (2004) pp.37-38

[4] Naoki Ohsugi. EASE Project: Introducing Empirical Software Engineering into Japanese Industry, International Workshop on Future Software Technology (IWFST) 2005, Shanghai, China, Nov.(2005)

[5] Yoshiki Mitani, Nahomi Kikuchi, Tomoko Matsumura, Satoshi Iwamura, Mike Barker, Ken-ichi Matsumoto:An empirical trial of multi-dimensional in-process measurement and feedback on a governmental multi-vendor software project. International Symposium on Empirical Software Engineering (ISESE) 2005, Vol.2, Noosa Heads, Australia, Nov. (2005) pp.5-8

[6] Yoshiki Mitani, Nahomi Kikuchi, Tomoko Matsumura, Satoshi Iwamura, Yoshiki Higo, Katsuro Inoue, Mike Barker, Ken-ichi Matsumoto: Effect of Software Industry Structure on a Research Framework for Empirical Software Engineering: 28th International Conference on Software Engineering(ICSE2006), Far East Experience Track, Poster Session; Shanghai, China, May (2006) pp.616-619

[7] Nahomi Kikuchi: Experience from Analysis of 1000 Collected Project Data, International Workshop on Future Software Technology (IWFST) 2005, Shanghai, China, Nov.(2005)

[8] SEC: http://www.ipa.go.jp/english/sec/index.html

[9] Naoki Ohsugi Akito Monden, Shuuji Morisaki: Collaborative Filtering Approach for Software Function Discovery: International Symposium on Empirical Software Engineering (ISESE) 2002, vol.2, Nara, Japan (2002) pp.45-46

[10] Naoki Ohsugi, Masateru Tsunoda, Akito Monden, Ken-ichi Matsumoto: Effort Estimation Based on Collaborative Filtering: 5th Intl. Conf. on Product Focused Soft. Process Improvement (PROFES) 2004, Nara, Japan (2004) pp.274-286.

[11] Toshihiro Kamiya, Shinji Kusumoto, Katsuro Inoue: CCFinder: A Multi-Linguistic Token-based Code Clone Detection System for Large Scale Source Code. IEEE TSE 28 (2002) pp.654-670

[12] IPA/Software Engineering Center (SEC): Software Development Data White Paper 2005: NIKKEI BP, 2005-5(2005) p137 (in Japanese)

[13] Naoki Ohsugi, Masateru Tsunoda, Akito Monden, Tomoko Matsumura, Ken-ichi Matsumoto, Nahomi Kikuchi: Using Cross-company Data to Estimate the Effort of Software Development Projects; SEC journal No.5 2006-2(2006) (in Japanese)

Exploring the Correlation between Predictive Usability Measures and User Tests

M.K.Donyaee[1], A.Seffah[1], And J. Rilling[1]

[1] Human-Centered Software Engineering Group,
Dept. Of Computer Science and Software Engineering,
Concordia University, Montreal, H3G 1M8, Canada
{donyaee, seffah, rilling}@cse.concordia.ca

Abstract. For software products to succeed they have to satisfy customer expectations, including usability aspects of the software. Early during the development cycle important design decisions are made for projects that are typically on restricted schedules and budgets. There is a need to support this decision making process, by empowering the decision makers with better tools and techniques. In this research we present a prediction approach that allows potential stakeholders to examine and choose among different user interfaces based on our usability prediction model. The usability predication model establishes correlations between predictive usability metrics and the results of usability tests performed by users. Based on our case studies we applied a multiple regression method to find relationships between response usability quality attributes and explanatory usability metrics.

Keywords: Usability, Software quality, Software metrics, Software measurement, Predictive model.

1 Introduction

Boehm has stated that one of the highest risks in software production is usability, commonly characterized by the software user interface [1]. This implies the need for considering usability of a software design as a factor in risk analysis as of the early stages of software life cycle. Over the past few decades or so several conceptual models have been proposed to measure usability. But just a few of them have the predictive power. And almost no model provides a usability score as a whole. Every model assesses just some aspects of it.

Predicting usability of a design as early as possible will help to reduce the number of development iterations. In the presented research we propose a solution to predict this quality aspect of software product from the metrics that are available at requirements or design stages. Our research will address the following question:

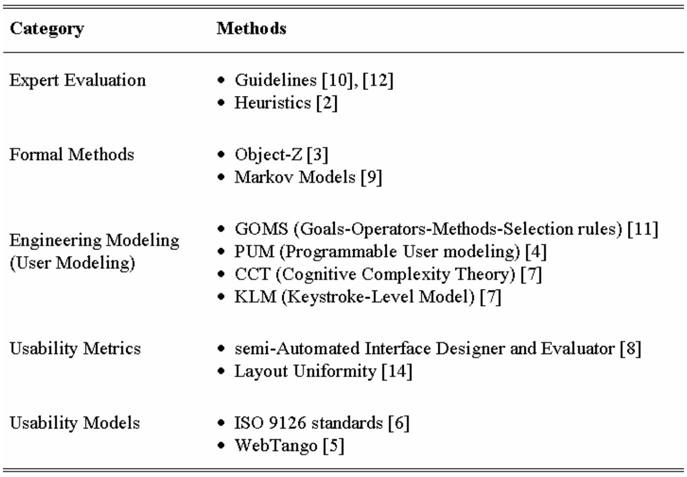
"What are the correlations between predictive usability metrics and the results of usability tests performed by users?"

The present paper is organized as follows. Section 2 will discuss related work with respect to usability predication. Section 3, will present the methodology on which our usability prediction models are based. Section 4 presents the data analysis and we present the results and discussion in section 5. We argue the limitations of our models in section 6 and finally Conclusions and future work are presented in section 7.

2 Related Work

In this section we introduce a classification of current usability prediction methods and analyze their impacts, their advantages and inconveniences during system development. Table 1 shows the categories and their respective methods. We would like to recall that the list of methods is not exhaustive and it just presents examples in each category.

Table 1. Usability prediction methods



Category	Methods
Expert Evaluation	• Guidelines [10], [12] • Heuristics [2]
Formal Methods	• Object-Z [3] • Markov Models [9]
Engineering Modeling (User Modeling)	• GOMS (Goals-Operators-Methods-Selection rules) [11] • PUM (Programmable User modeling) [4] • CCT (Cognitive Complexity Theory) [7] • KLM (Keystroke-Level Model) [7]
Usability Metrics	• semi-Automated Interface Designer and Evaluator [8] • Layout Uniformity [14]
Usability Models	• ISO 9126 standards [6] • WebTango [5]

So far the most commonly used category for the prediction of software usability has been expert evaluations, like guidelines and heuristics [2, 10, 12]. The basic characteristic common to this category is that they are easy to understand by software developers. Expert evaluations typically provide only a set of rules in plain text format to guide good software usability practice. However, the problem is that there are often too many of these heuristics and rules in each of the expert evaluation methods. Sometimes these rules in the evaluations are contradictory and ambiguous and therefore they may still require an expert in this field to complete the prediction process.

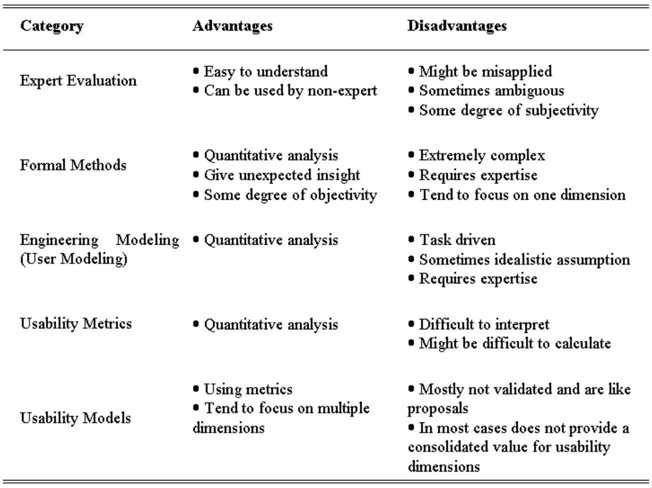
It is well known that applying formal specifications [3] in software development, with its ability to formalize, reason and verify may lead to significant higher costs. The cost is associated with the significant effort (time) required by the need for training of requirement analysts in the use of formal techniques and to write the formal specifications. There are also limitations of formal methods with respect to their ability to scale to large projects. Therefore limiting the applicability of formal specification to mostly safety critical systems and making them less suitable for non safety-critical systems.

Engineering models or user modeling systems provide a wide range of methodologies with the general goal to predict human behaviors. In some cases these methods are also referred to as task-modeling systems. Task-model driven systems, predict an individual aspect of a task, e.g. GOMS (Goals-Operators-Methods-Selection Rules) and KLM (Keystroke-Level Model). These task model driven systems are applied to predict a task execution time, CCT predicts the learning time for the user interface based on a novice user behavior [7]. Furthermore, they provide quantitative results that enable the comparison of different designs for a given task setting. However, these model driven systems are based sometimes on idealistic assumptions. For example GOMS assumes for predicting the task execution time that users do not make any mistakes [11]. These models also require additional expertise for identifying, setting up the tasks and analyzing the results and therefore causing additional cost.

Usability metrics have been shown to be useful in providing an objective insight with respect to the usability of tools in various different contexts. Some metrics are able to evaluate the system after development. Those are not in our interest. We may categorize the predictive usability metrics into two groups:

1. Task analysis metrics
2. User interface prototype metrics

Table 2. Advantages and disadvantages of methods



Category	Advantages	Disadvantages
Expert Evaluation	• Easy to understand • Can be used by non-expert	• Might be misapplied • Sometimes ambiguous • Some degree of subjectivity
Formal Methods	• Quantitative analysis • Give unexpected insight • Some degree of objectivity	• Extremely complex • Requires expertise • Tend to focus on one dimension
Engineering Modeling (User Modeling)	• Quantitative analysis	• Task driven • Sometimes idealistic assumption • Requires expertise
Usability Metrics	• Quantitative analysis	• Difficult to interpret • Might be difficult to calculate
Usability Models	• Using metrics • Tend to focus on multiple dimensions	• Mostly not validated and are like proposals • In most cases does not provide a consolidated value for usability dimensions

Task analysis metrics consider cognitive processes from a user perspective to address issues like learning time. Learning time can be predicted by the use of CCT (Cognitive Complexity Theory) method [7]. Prototype metrics like the Horizontal Balance [8] deal with graphical aspects that are visible on the screen. They attempt to predict the physical effort required by users to perform a specific task, or the ease a users can find an item on the screen. The difficulty in using such metrics is their interpretation. They typically only measure and therefore allow only to interpret one specific usability aspect. More over, even if there are multiple metrics available, no common framework exists on how to deduce a consolidated, more global evaluation and prediction of usability from these metrics.

Usability models are proposing a hierarchical relationship between different usability aspects and metrics, and they tend to assess the usability on multiple dimensions [6]. For example ISO standard 9126-4 tries to assess the safety aspect by relating it to a few metrics. The advantage of this method is having the objective assessment. But the model does not provide a method to assess safety as a consolidated quantitative value. Then it is up to the user of model after calculating the metrics, on how to interpret the metrics individually.

Webtango is a project that intends to create models to evaluate websites. The models predict the usability of a website from metrics. The results of the study shows that usability prediction with the help of metrics in some cases up to 80% matches the results of experts evaluation [18].

Table 2 presents the advantages and disadvantages of each category of usability prediction methods in a nutshell. The table is compiled from Nielsen [16] and Lansdale and Ormerod [17].

3 Research Methodology

In this research we adapted a deductive (top-down) approach for our reasoning. The overall goal of this research is to present prediction model for the global usability of a software system that can be applied before the actual implementation of a system. We defined our high level null hypothesis as the following:

“As the result of changes in the values of predictive usability metrics of software, there will be no changes in user perception of the usability.”

We collected numerical data from our presented case study to find evidence that supports or refuses our hypothesis.

3.1 Dependent Variables

The main dependent variable in our research methodology is the variable Global Usability. This variable specifies how user likes the system and how user is satisfied with using the system. We define Global Usability as [22]:

“Global Usability is a general indicator of goodness of a product from a user perception.”

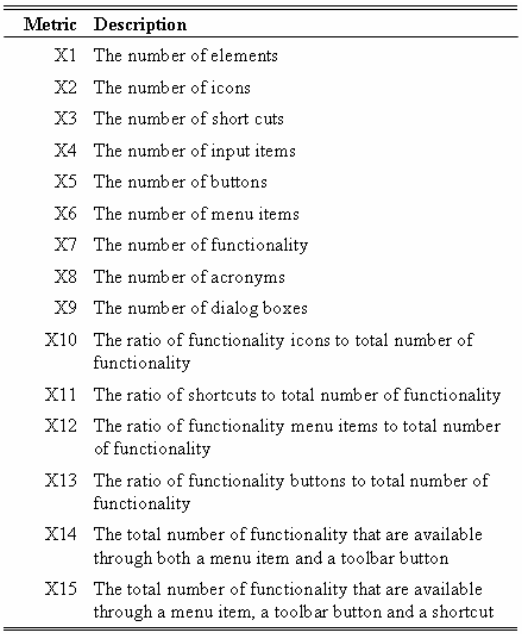
The other dependent variables used in our research that can represent either user satisfaction or a user’s perceptions of usability are: Affect, Efficiency, Helpfulness, Control and Learnability. These five variables correspond to the dimensions with which end users typically structure their judgment when they assess the usability of the software. These variables are measured by using SUMI [13]. SUMI (Software Usability Measurement Inventory) is a measurement program to assess user satisfaction. It consists of a standardized 50-items questionnaire and an extensive database built from data on a full range of software products. SUMI results have been shown to be reliable [19]. The variables are defined as follows [22]:

- *Affect*: This term refers to the user feeling mentally stimulated and pleasant or the opposite as a result of interacting with the software.
- *Helpfulness*: It refers to the user’s perceptions that the software communicates in a helpful way and assists in the resolution of operational problems.
- *Learnability*: This scale refers to the feeling of user that, it is relatively straightforward to become familiar the software.
- *Efficiency*: This variable refers to the user’s feeling that the software is enabling the task(s) to be performed in a quick, effective and economical manner or, at the opposite extreme, that the software is getting in the way of performance.
- *Control*: It refers to the user’s feeling that the software is responding in normal and consistent way to input and commands. It is not difficult to make the software work. The user can get the work done with ease.

3.2 Independent variables

All of the independent variables used in this research are based on existing user interface metrics. There exists a large body of research on usability metrics. From the large body of existing metrics we selected a subset that is applicable for our given experimental setting and goal.

Table 3. List of usability metrics



Metric	Description
X1	The number of elements
X2	The number of icons
X3	The number of short cuts
X4	The number of input items
X5	The number of buttons
X6	The number of menu items
X7	The number of functionality
X8	The number of acronyms
X9	The number of dialog boxes
X10	The ratio of functionality icons to total number of functionality
X11	The ratio of shortcuts to total number of functionality
X12	The ratio of functionality menu items to total number of functionality
X13	The ratio of functionality buttons to total number of functionality
X14	The total number of functionality that are available through both a menu item and a toolbar button
X15	The total number of functionality that are available through a menu item, a toolbar button and a shortcut

Some user interface metrics such as task execution time [11] are domain dependent and therefore would have limited our results to a particular task. Other metrics such as Layout Uniformity [14] are limited in their applicability due to the fact that today’s software products are highly customizable and therefore allowing the GUI and toolbars to be user customizable.

Table 3 shows the list of metrics that we applied for our prediction models. These metrics from Table 3 were selected based on their prediction power, corresponding to:

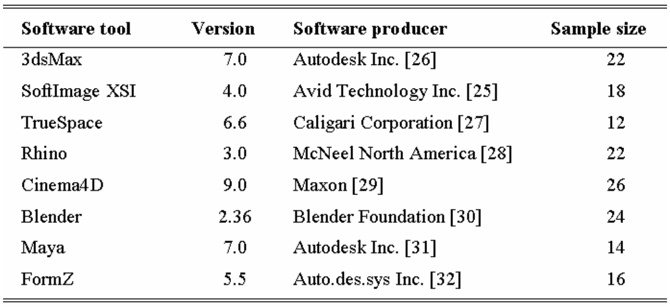
- Their ability to be determined from requirements or design artifacts.
- Their cost effectiveness with respect to time and effort that is required to collect the metric.
- Their applicability to different types of software applications, i.e. generality played an important role.

3.3 Experimental Design

We experimented with 8 COTS (Commercial-Off The-Shelf) software products that are used to create 3D graphical models. We chose to collect data in a specific domain in order to improve our comparison and analysis of the data collection. These complex software products have similar functionalities and their users are comparable with respect to their background and knowledge. However, the most important reason for selecting this application domain was that all of these tools have very sophisticated and complicated user interfaces. Therefore they are selected as candidate tools for this research. For every software product we collected the values of metrics. Table 4 shows the list of software applications that we used for the experiment.

For the dependent variables a minimum of 10 representative users is recommended in order to get accurate results with the SUMI assessment [21]. In our experiment settings we have a difference in sample size for every of our sample groups due to different factors such as availability and working experience with the product. But in all samples we have more than the recommended requirement by SUMI program. Table 4 reports the sample size for every group.

Table 4. List of software products



Software tool	Version	Software producer	Sample size
3dsMax	7.0	Autodesk Inc. [26]	22
SoftImage XSI	4.0	Avid Technology Inc. [25]	18
TrueSpace	6.6	Caligari Corporation [27]	12
Rhino	3.0	McNeel North America [28]	22
Cinema4D	9.0	Maxon [29]	26
Blender	2.36	Blender Foundation [30]	24
Maya	7.0	Autodesk Inc. [31]	14
FormZ	5.5	Auto.des.sys Inc. [32]	16

Before execution of experiments SUMI recommends to do a study on context of use to ensure that the context of use is the same for all users in our sample groups. The context of use study tries to answer three questions: Who is using the software, what are they doing with the software and where are they using the software. To collect data we randomly selected real users of every product around the world and asked them to give us information about their context of use. Subjects are chosen blindly through identifying their artistic works that are published on the Internet. Basically our sampling method is random without replacement and the sample size is determined based on SUMI requirements. In order to reduce the variability in sample data we controlled the experiment by asking subjects along with filling in the questionnaire, to provide other information about their experiences such as how many years of working experience they have with the target software. We also collected information about the hardware platforms that the target software was running on such as CPU speed and the amount of RAM.

Finally we built our sample population based on subjects with a minimum of one year working experience and using hardware platforms with a minimum of 512 Mb of RAM and CPU speed of 2.5 GHz or higher. Another control that we applied to our experiment was using the same version of software product by all subjects in the same sample group in order to reduce the risk of collecting bias data.

4 Data Analysis

Our usability scores are described using the medians of the data collected through the SUMI questionnaire. These medians were used as the outputs for the usability prediction models. By using the medians we tried to ensure that outlying results did not have too much impact on the data quality. We would like to recall that the outputs of the SUMI assessments are on a scale of 0 to 100 with a mean of 50 and a standard deviation of 10.

As for reliability of our collected data from the users, we used a 95% confidence interval for every score. This corresponds to the region where one can expect to find a true median of the sample 95% of the time, if one has a random replication of the study that was conducted. An interval value of less than 10 for us is an indicator that our data is well collected and users are coming from a homogenous group [22]. The only case in our data that conflicts with this rule is the score Affect. The confidence interval for Affect on some products is more than the valid value and it shows that not all users agree upon this score. We didn’t use this score in our analysis and modeling.

Testing of our hypothesis was done by selecting stepwise multiple linear regression modeling [23] as the analysis method. The reasons for selecting linear regression modeling were the following:

- We can control our independent variables, i.e. they are not random.
- We want to predict a value from another measured value.
- Our Y population has normal distribution, i.e. tested by normal probability plots [24].
- Both dependent and independent variables are continuous-valued variables.

- Individual polynomial curve fitting dose not show much difference between linear model and quadratic or cubic models.

The purpose of multiple linear regression is to establish a quantitative relationship between a group of predictor variables and a response. Such a model describes a hyper plane in the m-dimensional space of the regression variables $X_{m.}$ The parameter β_m represents the expected change in response y per unit change in X_m when all remaining regressors X_k ($k \neq m$) are held constants. Typically in linear models parameters are estimated by minimizing the sum of squares of residuals [23].
In order to test our null hypothesis we expressed it formally as:

$$H_0: \beta_1 = \beta_2 = _{\dots} = \beta_{15} = 0$$

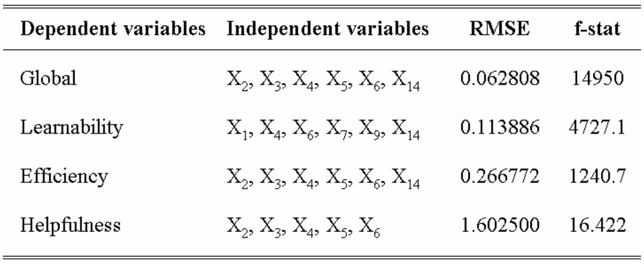
And For every dependent variable we started to build a multiple linear model by adding independent variables one by one to the model and evaluate the resulting statistics. The goal was to minimize RMSE (Root Mean Square Error) in the model [23].
We would like to recall that there exists also a well-known problem with the multiple linear regression method due to multicollinearity. Multicollinearity refers to interfering effects of some variables that can influence or obscure the values of other variables. This problem was addressed by applying a statistical tool called Partial Regression Leverage Plot (PRLP).

5 Results and Discussion

After the process of building regression models we determined what predictors have influence on what response variable. Based on the gathered information we refined our high level null hypothesis into more specific hypothesis:

$$\text{Global } H_0: \beta_2 = \beta_3 = \beta_4 = \beta_5 = \beta_6 = \beta_{14} = 0$$
$$\text{Learnability } H_0: \beta_1 = \beta_4 = \beta_6 = \beta_7 = \beta_9 = \beta_{14} = 0$$
$$\text{Efficiency } H_0: \beta_2 = \beta_3 = \beta_4 = \beta_5 = \beta_6 = \beta_{14} = 0$$
$$\text{Helpfulness } H_0: \beta_2 = \beta_3 = \beta_4 = \beta_5 = \beta_6 = 0$$

Table 5. The best-fitted regression models



Dependent variables	Independent variables	RMSE	f-stat
Global	$X_2, X_3, X_4, X_5, X_6, X_{14}$	0.062808	14950
Learnability	$X_1, X_4, X_6, X_7, X_9, X_{14}$	0.113886	4727.1
Efficiency	$X_2, X_3, X_4, X_5, X_6, X_{14}$	0.266772	1240.7
Helpfulness	X_2, X_3, X_4, X_5, X_6	1.602500	16.422

Table 5 shows the statistical measures that we determined for our best-fitted models. We were able to build models for response variables Global, Learnability, Efficiency and Helpfulness. In the attempts to build a model for variable Control we were not able to find any relationship between predictors and the response.

To test the significance of our regression models we used f-test [24] for every model. The determined values are reported in table 5. In the cases of Global, Efficiency and Learnability models since the f-test value at the 0.05 level of significance are all greater than the F critical value at this level which is 234.0, then we can reject the null hypothesis for those models. We formalize this as follows:

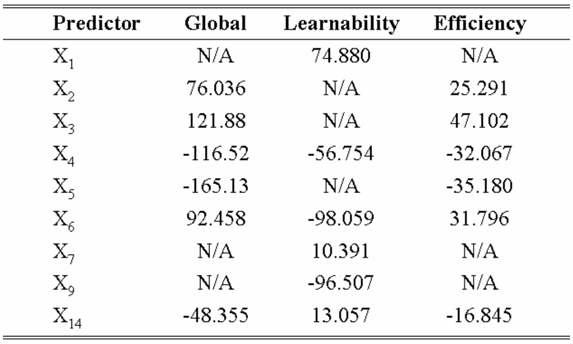
$$(\text{Global f-test} = 14950) > ({}_{0.95}F_{6,1} = 234.0)$$
$$(\text{Learnability f-test} = 4727.1) > ({}_{0.95}F_{6,1} = 234.0)$$
$$(\text{Efficiency f-test} = 1240.7) > ({}_{0.95}F_{6,1} = 234.0)$$

In the case of Helpfulness since the f-test value is less than the critical F value we are not able to reject the null hypothesis for this response variable:

$$(\text{Helpfulness f-test} = 16.422) < ({}_{0.95}F_{5,2} = 19.3)$$

The second test that we performed is the test of statistical significance of each partial regression coefficient, i.e. two-tailed t-test [24]. We need to perform this test since there is a possibility to have a statistically significant overall f ratio on the model itself and still some variables do not show a significant effect in prediction. Then by this test we evaluate the significance of predictors. Table 6 reports the t-test values calculated for predictors in models for Global, Learnability and Efficiency. In all cases the absolute value of t-test is greater than the critical value of T with 6 degree of freedom at the 0.025 level of significance that is 2.447. Therefore we are able to reject the null hypothesis (H_0: $\beta_i = 0$) for all predictors. That tells us each predictor X_i has statistically significant predictive capability in the presence of the other predictors.

Table 6. The results of t-test for individual predictors



Predictor	Global	Learnability	Efficiency
X_1	N/A	74.880	N/A
X_2	76.036	N/A	25.291
X_3	121.88	N/A	47.102
X_4	-116.52	-56.754	-32.067
X_5	-165.13	N/A	-35.180
X_6	92.458	-98.059	31.796
X_7	N/A	10.391	N/A
X_9	N/A	-96.507	N/A
X_{14}	-48.355	13.057	-16.845

6 Limitations

Although the statistical measures of our models show that they are significant in prediction, we believe that precaution has to be taken when using the models for the data outside of the range of data collected to build the model. We would like to recall that these models are tuned for the data we have collected. Then they are not necessarily working correctly for other data values.

Another limitation to our models could be the lack of generality. In our experiment design we have tried to take steps in a way that the models could be applicable to all types of software products. But this needs to be tested.

7 Conclusion

The motivation of our presented research was to provide a tool for software project stakeholders that can guide them during the decision process of predicting usability and selecting a user interface design among alternative user interface designs. The result of this research is of particular interest for software project managers, software developers, software engineers and user interface designers. Being able to determine and predict the overall usability of a system before implementation can provide significant cost reductions.

In our statistical analysis the significant results of regression analysis showed that the differences between samples are unlikely to be coincidence, or in the other word there is a high probability that differences we have observed between users perception of software usability are due to differences in the specific metrics.

Our literature review revealed that the research is the first of its kind. Our models are able to provide consolidated scores on Global usability, Learnability and Efficiency of a specific design. The quantitative scores enable us to compare different designs.

Finally we would like to express that as an extension to this research and future work we would like to collect more data to fine-tune the presented models. More over by trying the same experiment on other software categories we may test the generality of models.

References

1. Boehm, B.W., Software risk management: principles and practices, IEEE Software, Vol. 8, No. 1, 1991, pp. 32-41.
2. J. Nielsen, R. Molich, Heuristic evaluation of user interfaces, Proc. ACM CHI'90 Conf. (Seattle, WA, 1-5 April), 1990, pp. 249-256.
3. A. Hussey, I. MacColl, D. Carrington, Assessing usability from formal user interface designs, Proc. Software Engineering Conference, Australia, 2001, pp. 40-47.
4. Young, R. M., Green, T. R. G. and Simon, T., Programmable User Models for Predictive Evaluation of Interface Designs, Proceedings of CHI'89 Human Factors in Computing Systems, ACM Press, New- York, 1989.

5. Melody Y. Ivory, Rashmi R. Sinha, Marti A. Hearst, Empirically validated web page design metrics, SIGCHI'01, Seattle, WA, USA, ACM 1-58113-327-8/01/0003, 2001.
6. ISO 9126-4, Software Engineering, Software product quality, part 4 Quality in use metrics, 2001.
7. E. John Bonnie, E. Kieras David, Using GOMS for User Interface Design and Evaluation: Which Technique?, ACM Transactions on Computer-Human Interaction, Vol. 3, No. 4, 1996, pp. 287-319.
8. A. Sears, AIDE: A step toward metric-based interface development tools, UIST, Pittsburgh, PA, USA, ACM 1995 0-89791-709-x/95/11. Available: http://portal.acm.org/citation.cfm?id=215704&dl=ACM&coll=portal.
9. Cairns P, Jones M, Thimblery H, Usability analysis with Markov models, ACM Transactions on Computer-Human Interactions, Vol. 8, No. 2, 2001, pp. 99-132.
10. Smith, S. L. and Mosier, J. N., Design Guidelines fo Designing User Interface Software. Technical Report MTR-10090, The MITRE Corporation, Bedford, MA, 1986.
11. Card, S. K., Moran, J. P. and Newell, A., The Psychology of Human Computer Interaction, Erlbaum, Hillsdale, NJ, 1983.
12. Apple Computer Inc., Human Interface Guidelines: The Apple DesktopInterface. 2nd ed., Addison-Wesley, Reading, MA, 1992.
13. Kirakowski, J. and Corbett, M., SUMI: the software measurement inventory, British Journal of Educational Technology, 24, 1993, pp. 210-212.
14. L. L.Constantine, L. A. D. Lockwood, Software for Use, Addison-Wesley, 1999, pp. 417-442.
15. Han X. Lin, Yee-Yin Choong and Gavriel Salvendy, A Proposed Index of Usability: A Method for Comparing the Relative Usability of Different Software Systems, Behavior and Information Technology, 1997, Vol. 16, No. 4/5, pp. 267-278.
16. Nielsen, J., Usability Engineering, Academic, San Diego, CA, 1993.
17. Lansdale, M. W. and Ormerod, T. C., Understanding Interfaces: a Handbook of Human-Computer Dialogue, 1994, (Academic Press, San Diego, CA).
18. Y. Ivory Melody, R. Sinha Rashmi, A. Hearst Marti, Empirically validated web page design metrics, SIGCHI'01, Seattle, WA, USA, ACM 2001 1-58113-327-8/01/0003.
19. N. Baven, M. Macleod , Usability measurement in context, Behavior and Information Technology, 13, 1994, pp. 132-145.
20. Michael K. Daskalantonakis, A practical view of software measurement and implementaion experiences within Motorola, IEEE Transaction on Software Engineering, Vol 18, No, 11, Nov. 1992, pp 998-101.
21. E. V. Veenendaal,, Questionnaire based usability testing, Project Control for Software Quality, Proc. of ESCOM SCOPE 99, Herstmonceux, England, 1999.
22. J. Kirakowski, SUMI User Handbook, University College Cork, Ireland, 1998.
23. A. Hughes, D. Grawoig, Statistics: A Foundation for Analysis, Addison-Wesley Publishing Company, USA, 1971, pp. 362-380.
24. Richard G. Lomax, Statistical Concepts, Lawrence Erlbaum Associates, Inc., 1998, pp 48-86.
25. Avid Technolopgy, Inc., 2006, Avaiable: http://www.softimage.com/home/
26. Autodesk, Inc., 2006, Available: http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=5708192
27. Caligari Corporation, 2006, Available: http://www.caligari.com/
28. McNeel North America, 2006, Available: http://www.rhino3d.com/
29 Maxon, 2006, Available: http://www.maxoncomputer.com/
30. Blender Foundation, 2006, Available: http://www.blender.org/cms/Home.2.0.html
31. Autodesk Inc., 2006, Available: http://www.alias.com/eng/index.shtml
32. Auto.des.sys Inc., 2006, Available: http://www.formz.com/

FORESEEing for Better Project Sizing & Effort Estimation

Luigi Buglione

Atos Origin, Via Riccardo Morandi 32,
I-00050 Rome, Italy
Luigi.buglione@atosorigin.com

Abstract. Nonetheless the huge amount of technical literature available, sizing and estimation are often perceived as a sort of "divinatory art" more than an issue to scientifically manage. There are plenty of parametric estimation models (e.g. COCOMO, SLIM, …) applied by thousand of people but usually without verifying if and how much they fit with their organizations. This paper presents the goals and results from a pilot program named FORESEE (Functional Requirements for Sizing & Effort Estimation carried out by the Atos Origin's Italian System Integration Department during 2005 with the main objective to transfer a practical methodology for improving sizing & estimation practices in software projects.

Keywords: Effort Estimation, FORESEE, PSU, Functional Size Measurement Methods, Project Historical Database, Project Size

1 Introduction

One of the main critical issues in Software Project Management is the optimization of project estimations. And two main issues to analyze seem to be *project size* (whatever the nature of a project) and *historical project databases*.

About the first issue (*project size*), one of the most diffused weaknesses is the misapplication of project sizing measurements. In order to verify this affirmation, it is sufficient to take a look at the PMBOK2004 guide [9], one of the most used and referenced Project Management guides, observing that "*project size*" is a term introduced several times along chapters but never defined, neither within the text, nor in the glossary. Also taking a look at maturity models specifically devoted to the Project Management domain such as P3M3 (Portfolio, Programme and Project Management Maturity Model) [7][8] and P2MM (Prince2 Maturity Model) [11], project estimations are usually referred only to costs, with no direct links to the size of the project itself. Another issue is that using productivity ratios based on a single software size unit could not be sufficient to obtain properly approximated estimates, as recently discussed by Kitchenham & Mendez [5].

About the second issue (*historical project databases*), another usual difficulty is to continuously gather projects' data after the project is closed, designing an experience database containing those fields really valuable to the company for estimation purposes, integrated as more as possible with the Information Systems. The recent diffusion of external databases like the one by ISBSG [10] can help a lot in increasing the capability of an ICT company to build their own project databases.

Moving from this thoughts, in a continuous improvement strategy and in order to reinforce more and more the experience and value of its personnel devoted to size & estimate projects (not only Project Managers), the Italian Systems Integration Department of Atos Origin[14] committed in 2005 an improvement action on this issue, a pilot program called FORESEE (Functional Requirements for Size & Effort Estimation). Started in 2005 and involving closely 150 people, FORESEE objective was to transfer a practical methodology for sizing and estimating project effort from the bidding phase on with a "training & support-on-projects" approach.

The paper is organized as follows: Section 2 describes possible estimation paths, according to lifecycle phase and techniques to adopt. Section 3 introduces the FORESEE program, presenting the sizing techniques used, phases and assumptions for the program, as well as the project data gathering issue. Section 4 shows lessons learned and recommendations derived after the program ended. Finally, Section 5 draws conclusions and prospects for this initiative.

2 Possible Estimation Paths

2.1 Background

Figure 1 shows two possible estimation paths: the current one, shown on the left ("today") and what should be a target scenario ("tomorrow"), filling some common gaps.

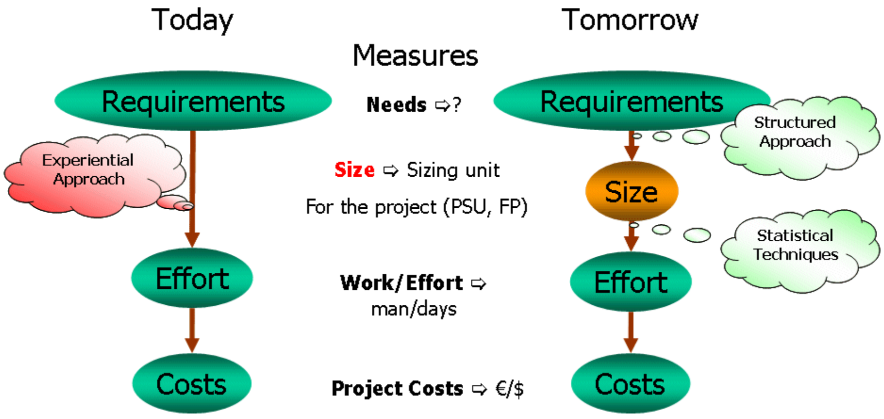


Fig. 1. Possible Estimation Paths: *today* and *tomorrow*

[14] Cfr. www.atosorigin.com (Corporate site); www.atosorigin.it (Italian site)

The current scenario ("*today*"), followe* by many organizations (no matter the number of employees and yearly turnover), presents an estimation process where effort is derived from the analysis of customer requirements applying an analogy or experiential approach. In this case, it is fundamental the role of the estimator and his/her experience in the field; a possible, significant benefit can be done if historical data from closed projects are constantly gathered. Some open issues can come from missing historical data and missing calculation of the estimation error both at the project level as well as at the organization level: MRE (Mean Relative Error), MMRE (Magnitude of MRE), Pred(0.x%), r, R^2 are just few statistical measures used for evaluating estimations [3].

The second scenario ("*tomorrow*") takes into account further information in a structured manner: the *project size*, to explicit how "big" is the whole project to be managed, not only a part of it. A Software Project Repository usually gather the functional size of software projects, expressed through a certain functional size unit (fsu) according to the functional size measurement (FSMM) method chosen and applied. But in those repositories the functional size is usually related to the overall project effort and not only to the functional-related part. Furthermore, usually there is no information about the approximate percentage between functional vs. non-functional effort that could help estimators in improving their estimates and have more issues for analyzing eventual errors. As Kitchenham and Mendez suggested the concurrent usage of more than one size unit for calculating the productivity (*AdjustedSize/Effort*) [5], also when speaking about effort estimation more than one size unit can be adopted along the different SLC phases for reducing as much as possible projects' MREs.

Finally, another improvement would be the introduction of more quantitative estimation techniques as well as an organizational standard process, using its own historical data as much as possible, in addition to the estimation by analogy or experience, that also well-known and recognized guides such as PMBOK (Project Management Body of Knowledge) suggest before the quantitative ones[15].

2.2 Early Estimations in the SLC context

Figure 2 shows a possible classification of stages (Bid; Delivery) and related phases for deploying a project. At the Bid stage, early FSM techniques are available in order to overcome the problem of the full application of a FSMM and could be put in action, but few companies decide to spend at this stage the amount of time and effort needed for performing a functional analysis.

Again, at this stage the depth of customer requirements probably is not sufficient to properly calculate a number of functional units quite close to the one that will be calculated at the end of the Analysis phase, therefore not achieving the goal to anticipate the functional size with an acceptable error margin.

Last but not least, a project manager at the Delivery stage has to establish the project plan under the heritage of estimates done at the Bid stage. Therefore

[15] See [9], Chapter 6.4 (*Activity Duration Estimations*).

estimating the size of the whole project, having approximately the idea of which is the size of the whole project (not only the size of its work products) as well as the ratio functional vs. non-functional effort, can allow a better communication between the bid manager (and/or an estimation expert) and the project manager in different stages along the project lifetime.

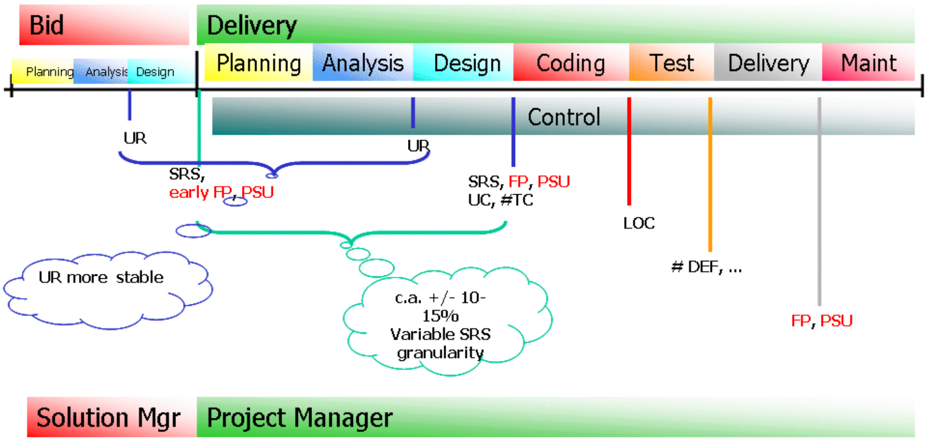


Fig. 2. Estimation and Software Life Cycle (SLC): phases and possible sizing units

For instance, some TLC enhancement projects in the pre-paid line revealed that the percentage of the effort from the deployment of functional requirements was averagely of 30-35% of the overall project effort, while MIS projects had the opposite balancing. Therefore, in order to minimize projects' MRE, we should use also this info in the project historical database as a further filter.

Thus, a series of questions come out, where the solely management and usage of a FSM probably cannot answer: How much is large your (whole) project? How can I estimate the non-functional size of next projects? And what about service management projects? Which size unit could I use?

In conclusion, an emerging need is to measure not only the *product* and the *process* (through some FSMM-based ratio [17]), but also the *project* and understand which relationships link the two entities and how to manage them in order to obtain better estimations during time.

3 The FORESEE Program

3.1 Background and Program Goals

The FORESEE (Functional Requirements for Sizing and Effort Estimation) program was the improvement action taken by Atos Origin. The acronym and the logo presented in Figure 3 born from the willing not only to “SEE” (Sizing & Effort Estimation) in the short term, but to have a longer view see beyond (“FORESEE” – or “4SEE”, using a phonetical joke) this short term, gathering project historical data and increasing the ability of estimators or anybody else in an organization interested to the estimation issue through the usage of more sizing measures.

Fig. 3. FORESEE Logo

Two main goals to achieve:

- Support the estimation process, currently based on an experiential-analogy approach, in order to obtain a better adherence of estimated efforts against the actual ones.
- Develop the needed skills in order to satisfy the needs of specific customers, improving and better qualifying our ranking in their eyes.

through

- The introduction of standardized and objective techniques, supporting project sizing as the main input for the subsequent effort estimation process.
- The introduction of a relationship, across different SLC phases, between (at least) two techniques, an early (*estimation*) and a full (*counting*) one.
- The improvement of projects’ historical database, shared and accessible to people making estimates in projects.

The program was conducted during 2005 and addressed Pilot Areas from each Atos Origin delivery structure and from each one of the Systems Integration Italian sites (Milan, Naples, Rome and Turin), involving closely 150 people. The roles typically interested were those usually involved in management activities strongly based on estimates:

- Solution Managers[16]
- Project Managers
- Analysts
- Architects
- Team Leaders

The approach takes into account an application and gradual extension of the program, through the continuous involvement of well identified areas within Systems Integration Business Structures.

[16] This role expresses the technical experts involved during the Bid stage for proposing the technical solution that will optimize customer’s requests.

3.2 Sizing Techniques

As said above, at least two sizing techniques were chosen: one for early estimation, Project Size Unit (PSU) [2][17], and one for a full count, typically a FSMM, in this case IFPUG FPA v4.2 [4]. Next tables propose two different comparisons between PSU and FPA in order to stress their complementary usage within an ICT organization.

Table 1. FPA and PSU: comparing the technique structures.

Method\ Elements	*Application Domain*	*Entity*	*Complexity*	*Adj. Factor*	*Complexity*
FPA* (standard*)	*Software Engineering*	*Data (ILF, EIF) and Transactions (EI, EO, EQ) related to the functional requirement for a software system*	*3 levels (H/M/L) for each type of entity.*	*14 General System Characteristics (GSC)*	*Weight (0-5) for each one of the 14 GSCs, with a ±35% variability on the value of unadjusted FPs*
PSU* (early*)	*General Purpose*	*Detailed UR (not only functional) and derived tasks (rule: 1 task = max* x *m/d) for the technical tasks of the SLC.*	*3 levels (H/M/L) for each detailed tasks per UR (and therefore of m/d, statistically derived).*	*% weighting for evaluating the amount of effort needed in proportion for management and qualitative tasks.*	*This percentage is derived from the analysis of the historical project database and it is proportional to the number of unadjusted PSU.*

Table 2. FPA and PSU: a more detailed comparison.

	Early methods (PSU)	*Standard methods (FPA)*
SLC phase to be applied	*Planning (level L-1)*	*Design (level L-2)*
Accuracy level	*Lower than standard methods (avg)*	*Greater than "early" methods (avg)*
The size unit refers to	*Project*	*Functional User Requirements (FUR)*
Control parameters for verifying the estimation accuracy	In both cases, MRE and Pred(0.25) values calculated on the estimated effort must be compared with those calculated at the end of the project and the MMRE and Pred(0.25) on the entire set of projects included in the historical project database used for the forecasting system.	
Information level needed	*Documentation from the Bid phase*	*Documentation from the Analysis phase*
Requested skills for estimation	*Project team*	*Function Points Counter (preferably a CFPS)*
Time needed for estimation	*0.5 m/d (per PSU counting)*	*1.5-2 m/d (per FPA counting for medium-size projects [15])*
Strengths	• *Quick calculation* • *Not requested FPA knowledge* • *Project estimation can be done before the Analysis & Design phase*	• *Greater accuracy in the size calculation to be used for the estimation* • *External comparability of results*

[17] See also http://www.geocities.com/lbu_measure/psu/psu.htm

Weaknesses	- *Lower accuracy in the size calculation to be used for estimation, verification of correlation with "standard" techniques* - *Internal Comparability of results*	- *Greater effort for deriving the number of FPs* - *Requested the knowledge of FPA* - *Project Estimation can be done before starting the Developing phase (Coding)*
Comments	*Experimental & Internal technique.*	*Consolidated and diffused technique, with counting rules regularly monitored by International bodies.*

Joining the two techniques allows to overcome some inner limits of applying those techniques one at a time. The application of those measures (PSU and FP), properly traced in the PHD (see next section) during the whole project lifecycle will allow to determine the relationship between the two measures, at the aim to estimate the number of FP (FP*) through the number of (counted) PSU in the Bid phase. The effectiveness of such approach, based on statistical techniques, will be directly proportional to its continuous, extended and coherent application throughout the organization [16].

3.3 PHD: Project Historical Database

None of the proposed techniques can provide the expected benefits without the usage of a consistent project historical database (**PHD** - Project Historical Database). Thus, one of the program goals was to improve the structure of such database gathering projects' data (on-going or closed projects) involved in the Pilot Program.

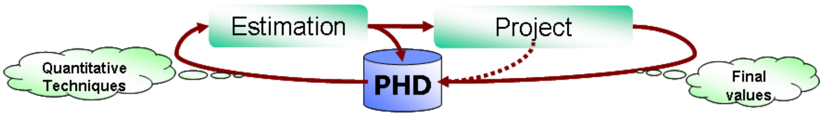


Fig. 4. Project Historical Database (PHD) and its usages

As in well-known and recognized software projects repositories such as ISBSG r9 [14], PHD contains a series of fields referred to organizational and technical issues. Sizing data (and related efforts), expressed with at least two measures (PSU and a FSMM), were gathered in three times: at the Bid stage and, at the Delivery stage, after the Design and Delivery phases, allowing analysis for detecting possible causes for estimation errors.

3.4 Program Assumptions

In order to maximize the informative return from the program, before the training sessions started it was asked to each participant (or couple of participants, if working on the same project line) to send to FORESEE team at least some data/documents from one past project they where worked in. In this way, it was possible to work on real case studies, more interesting to participants and having real data to populate PHD. These projects where marked with an identifier, in order to do not mix "backfired" sizing data with those ones from new projects. In particular the information-documentation required before attending the training was:

- Project Plan & Quality Plan, for acquiring information on project background and organization, useful to populate PHD.
- Effort (m/d), estimated and final, in order to have absolute values and derive the estimation error figures.
- Gantt or WBS (final version), in order to look at estimates with a project management approach as the main input for calculating PSU.
- User Requirements & Software Requirement Specifications, as inputs for FPA calculation.

3.5 Program Phases

The program was structured into six phases (four "basic" and two "support" ones), as shown in Fig.4.

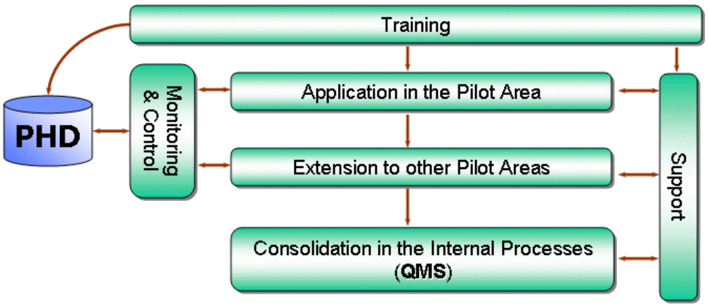

Training
PHD
Monitoring & Control
Application in the Pilot Area
Extension to other Pilot Areas
Consolidation in the Internal Processes (QMS)
Support

Fig. 5. FORESEE Program phases

The "basic" ones are:

- **Training**. Five modules: Introduction to Project Sizing & Estimation (0.5m/d), Project Size Unit (2.0m/d), Introduction to Functional Size Measurement (0.5m/d), IFPUG FPA v4.2 (3.0m/d), Effort Estimation in Software Projects (2.0m/d). Training sessions were strongly based on practical exercises on real case studies, with a maximum number of 7-8 participants, in order to have sufficient time to provide support and answering questions during the day. Exercises on PSU technique allowed to populate PHD (Project Historical Database), while exercises on FPA allowed partial counts from the selected projects; total counts were managed by Pilot Areas people in the "Application" phase.
- **Application in the Pilot Area**. After completing the training, those techniques (PSU, FPA, Effort Estimation using PHD) are applied to new projects.

- **Extension to other Pilot Areas**: Based on the "*butterfly effect*"[18], the program was extended to more Pilot Areas and projects starting to use PHD for estimates, replicating the same working framework proposed.
- **Consolidation in the internal processes (QMS)**: it is one of the outcomes of the program: to "encapsulate" experience from this improvement program into the local Quality Management System (QMS).

During the Pilot period, two-part time resources from the FORESEE team provided "support" in term of:

- **Monitoring & Control**: while pilot projects are going on, there is a supplementary Quality Assurance activity in place about the verification & validation (V&V) of data to be inserted into the PHD, before making those data public within the organization.
- **Support**:
  - A methodological help-desk directly to Pilot Projects teams about the proper and correct application of those techniques and methods;
  - Feedbacks to the Quality Department for embedding the FORESEE experience and updating the software development & maintenance processes of the Italian QMS.

4 Lessons Learned & Recommendations

After conducting four pilot sessions, an analysis of feedback returned and gathered was conducted. In the following paragraphs, main figures and results are presented.

Table 3. FORESEE: main figures about attendees and feedback received.

Training Modules	# invitations (A)	# participants (B)	% (B/A)	Feedback (C)	% (C/B)
1 – Intro & SwMeas.	93	76	81,72%	59	63,44%
2 - PSU	34	30	88,24%	20	75,00%
3 - IFPUG FPA Intro	94	62	65,96%	43	45,74%
4 - IFPUG FPA v4.2	34	24	70,59%	18	75,00%
5 - Effort Estimation	34	21	61,76%	14	66,67%

Here the main strengths from the analysis of feedbacks received in a qualitative form:

- **Satisfaction of Expectation from the program**: Medium-high level ratings, consistent throughout all modules and sessions
- **Clarity of Contents**: Each comment rated as not completely positive generated a "change request" to be implemented before next program pilot session starts

[18] http://www.cmp.caltech.edu/~mcc/chaos_new/Lorenz.html

- **Modules Rating**: average "Good" rating[19] close to 60% for all the five modules
- **Suggesting FORESEE to other colleagues**: quite all participants would suggest to other colleagues to participate to next FORESEE sessions.

And main improvements issues to implement for the future:

- **Everyday work advantages**: modules with the lowest ratings where Module #1 (introduction to the program and to Sw Measurement) and Modules #3-4 (IFPUG FPA). Respondents not familiar with the technique need more practice on counting rules applied to internal projects.
- **Evaluate the opportunity to schedule training modules on half working days**: some respondents suggested to provide training splitting whole working days in two shorter ones, allowing them to bring out their everyday work. But this proposal will be weighted with the expected information effectiveness, module by module
- **Divide informative and formative paths**[20]: the goal is to avoid some information for those participants attending all the five training modules.

Some further evidences and results:

- Started a "backfiring" campaign for project managers, in order to retrieve the higher number of data points from closed projects, taking a look in particular to those kind of project with a short duration (e.g. TLC enhancements, web portals, …). Some advantages: an estimator can start being familiar with PHD, building a part of this organizational "history", being part of it yet with old projects data and stimulating other colleagues to do the same, generating the "butterfly effect" previously mentioned.
- After the program was concluded, PHD included 30 new projects' data. Averagely each pilot introduces at least "backfired" data from 3-4 projects and 2-3 new ones after the training has been completed.
- PHD structure was improved, adding new fields suggested by practitioners (e.g.: number of average FTE, Duration time, …)

5 Conclusions & Prospects

The usage of Functional Size Measurement Methods (FSMM) has grown a lot during last 10 years and also ISO created a specific working group (WG12) for managing this issue. But project estimates cannot be reduced only to a functional sizing for a series of reasons: we need to know and measure a more comprehensive entity (project) understanding relationships with product and process-related measures (such as Function Points) as well as issues an FSMM cannot measure (non-functional tasks), in order to obtain better estimations and minimize as much as possible errors.

[19] On a four-level rating scale (Weak, Fair, Good, Excellent).

[20] The informative path is given by Modules #1 and #3, as an introduction to these issues, while the formative path is given by Modules #2, #4 and #5.

Atos Origin, in the logic of continuous improvement, committed in 2005 an improvement program devoted to increase the internal knowledge and application of sizing methods for improving the quality of estimates in software projects, named FORESEE (Functional Requirements for Sizing & Effort Estimation). This paper provided the program background, outlines, feedbacks and main improvements and strenghts.

Core elements of the FORESEE program were:

- Complementary usage of two sizing metrics, an early one (PSU v1.01) [2] and a "standard" one (IFPUG Function Point Analysis v4.2 [4], right now the most applied and requested FSMM both by Italian private companies and Public Administrations). About this issue, the level of analysis details available in the bidding phase cannot always be sufficient to fully take advantage from a FSMM for estimation purposes as well as after the Design phase. Again, some systems have a relevant percentage of non-functional requirements (i.e. some enhancements projects for TLC systems), needing further sizing measures to complement functional size measures.
- The estimation approach based on regression analysis with previous projects' quantitative data from the Project Historical Database (PHD), evolved from previous Italian Sw-CMM experiences: the "as-is" usage of "external" cost estimation models such as COCOMO [1] or SLIM [12] should be avoided, because it can lead to relevant project MREs if not properly adopted. And right now it seems there are no studies providing figures about the cost for calibrating the model, that's an informative input essential to an evaluator for deciding (or not) if applying such model at certain error ranges [3].

The consolidation of pilots' results (processes, templates, tools) represented an input for periodical updates of the Italian Quality Management System (QMS) processes, ISO 9001:2000 certified from 1996 and compliant with Sw-CMM ML3 practices.

Next step for the FORESEE program application will be the extension of principles and application also to IT service projects, properly customizing the basic concept yet adopted within the Systems Engineering domain and sizing projects with PSU. In fact, also in specific SPI models for IT services as the IT Service-CMM [6], service "estimation" is provided at Level 2 using historical data but with no explicit activities for measuring the "size" of service work products, introduced only at Level 3 in the Integrated Service Management KPA. In this case, the main challenge will be to design the right project fields and customize the PHD structure in order to be useful for IT service projects.

References

1. Boehm B.W., Horowitz E., Madachy R., Reifer D., Clark B.K., Steece B., Winsor Brown A., Chulani S. & Abts C., Software Cost Estimation with COCOMO II, Prentice Hall, 2000, ISBN 0130266922

2. Buglione L., Project Size Unit (PSU) - Measurement Manual, v1.01, October 2005, URL: http://www.geocities.com/lbu_measure/psu/psu.htm
3. Conte S., Dunsmore H. & Shen V.Y., Software Engineering Metrics and Models, Benjamin Cummings: Manlo Park, CA, 1986, ISBN:0-8053-2162-4
4. IFPUG, Function Points Counting Practices Manual (release 4.2), International Function Point User Group, Westerville, Ohio, April 2000, URL: http://www.ifpug.org
5. Kitchenham B. & Mendes E., Software Productivity Measurement Using Multiple Size Measures, IEEE Transactions on Software Engineering, Vol.30, No.12, December 2004, pp.1023-1035; URL: http://www.cs.auckland.ac.nz/~emilia/publications/TSE2004.pdf
6. Niessink F., Clerc V., Tijdink T. & Van Vliet H., The IT Service Capability Maturity Model, Version 1.0, Release Candidate 1, Jan18 2005, URL: http://www.itservicecmm.org
7. OGC, Project Management Maturity Model, Version 5.0, October 2002, URL: http://www.ogc.gov.uk/sdtoolkit/reference/tools/PMMM_release_v5.pdf
8. OGC, Portfolio, Programme & Project Management Maturity Model (P3M3), Version 0.1, December 2003, Office of Government Commerce, URL: http://www.ogc.gov.uk/sdtoolkit/reference/tools/p3m3.pdf
9. PMI, A Guide to the Project Management Body of Knowledge (PMBOK), 3/e, Project Management Institute, ISBN 1-930699-45-X, 2004, URL: http://www.pmi.org
10. ISBSG, ISBSG R9 Database, International Software Benchmarking Standards Group, 2005
11. OGC, Prince2 Maturity Model, Version 3.0, Draft, April 2004, URL: http://www.prince2.org.uk/nmsruntime/saveasdialog.asp?lID=788&sID=241
12. Putnam, L.H. A general empirical solution to the macro software sizing and estimation problem. IEEE Transactions on Sofiware Engineering, vol.4, no. 4, July 1978, pp.345-381
13. Buglione L., Dimensionamento dei progetti: che 'metro' usare?, XPM.it, June 2006, URL: http://www.xpm.it
14. ISBSG, *ISBSG R9 Database,* International Software Benchmarking Standards Group, 2005, URL: http://www.isbsg.org
15. Santillo L., Lombardi S., Natale D., Advances in statistical analysis from the ISBSG benchmarking database, Proceedings of SMEF2005 (2nd Software European Measurement Forum), Rome (Italy), 16-18 March 2005, URL: http://www.dpo.it/smef2005/filez/g105_gufpi.pdf
16. Buglione L., *PSU e Metriche Funzionali per il Dimensionamento del Software: concorrenti o alleati?,* White Paper, Bloom.it - Frammenti di Organizzazione, 14 Marzo 2005, URL: http://www.bloom.it/buglione2.htm
17. IFPUG, Guidelines to Software Measurement (release 2), International Function Point User Group, August 2004, URL: http://www.ifpug.org

Complexity and Development Effort Prediction Models Using Component Oriented Metrics

Nael Salman and Ali H. Dogru[1]

Computer Engineering Department,
Cankaya University, Ankara, Turkey
Email: nsalman@cankaya.edu.tr
[1]Computer Engineering Department,
Middle East Technical University, Ankara, Turkey
Email: dogru@ceng.metu.edu.tr

Abstract. Complexity and development effort estimation models are developed based on a set of Component Oriented metrics. Our view focuses on the efforts for the identification and the integration of existing components, as the most important types of development activities in this emerging paradigm. The metrics set that is used in this research, therefore, is specifically configured considering the composition of related elements of a Component Oriented design. Regression models were obtained using data collected from student projects. The obtained models were validated using data collected from 5 projects. The results of the validation show that Component Oriented metrics can be used by developers and managers in predicting product and process related parameters.

Keywords: component orientation, complexity, development effort, metrics.

1 Introduction

Component Orientation (CO) is the new trend towards software development which focuses on building large and efficient software systems by integrating prefabricated components [4, 7]. The most important types of development activities in this emerging paradigm are the identification and integration of existing components. CO uses units of abstraction which are different from those used in the object oriented and the earlier paradigms. In CO main focus is on “structure” contrasted to “data” in object orientation and “function” in the traditional paradigm [4, 10]. Gorla and Ramakrishnan [5] defined software system structure as the way through which system building elements are organized in relation to each other and their environment. Based on that view, a new orientation in the metrics set is necessary because the CO paradigm does not suggest “code writing”, thus displaying a radical difference from any previous software development approach.

A literature review on CO reveals that little effort has been dedicated to metrics characterizing components and CO systems. Basili and Boehm [2] emphasized the need for metrics characterizing component customizability, integrability, and reliability. Vitharana et al. [9] showed that cost estimation, productivity estimation, reliability, and maintainability measurements are still open research fields for component oriented software engineering.

2 CO Metrics

This research extends the previous work on Component Oriented metrics [11, 12] that was inspired by the Object Oriented metrics work widely known as CK metrics [13, 14]. Structural aspects of CO software systems that may influence the product size, and/or the process are identified as: components, connectors, and the composition hierarchy.

Metrics quantifying these aspects are defined as follows:

1) Total Number of Components (TNC): The count of all components in the system. This includes both abstract and implemented components. Total number of components in the systems is a design decision. While some designers may favor relatively smaller components, another design decision may favor a fewer number of relatively larger components. So the total number of components in the system is a design issue that influences the overall structure of the system.
2) Depth of Composition Tree (DCT): Count of the number of distinct levels of the composition tree. This metric is identified based on the following initial viewpoints:
   a. The deeper the composition tree the better the system decomposition is. Higher values of DCT are an indication that system components are more specific and may have higher potentials for inter-system reuse.
   b. The deeper the DCT the more components we have. Components, at levels closer to the root of the tree tend to be having many sub-components making them more difficult to compose and test.

3) Total Number of Connectors (TNCO): The sum of the total number of abstract connectors plus the total number of messages. This metric is a measure of the degree of coupling in the system.
4) Total Number of Interfaces (TNI): The count of the number of interfaces of all components in the system. Number of interfaces per component is an indication of the amount and diversity of functionality delivered by the components. This metric is identified based on the following viewpoints:
   a. Increased number of interfaces implies a wide set of services since an interface usually presents one category of services for a component. This may limit the possibilities of a component reuse among systems as also, diversified interfaces indicate specialized connections.
   b. Increased number of interfaces implies increased fan-in value for a component which means that it is supposed to be highly dependable; it should be designed, implemented, and tested with a lot of care. A bug that

may exist in such a component may be cascaded to several other dependent components.

c. On the other hand, increased number of interfaces of a component could mean clear service descriptions and then would lead to less effort to integrate with other components.

5) Total Number of Methods (TNM): The count of the total number of methods in the system. Components implement their functionality in their methods. More methods in the system indicate increased functionality. More methods in the system indicate increased functionality. Two systems that deliver the same functionality where one has less number of methods indicate that methods of the system represent a wider range of services.

6) Total Number of Implemented Components (TNIC): The count of the total number of implemented components only. Implemented components are where system functionality is implemented.
7) Average Number of Methods per Components (ANMC): The ratio of the total number of methods to the total number of components in the system.
8) Average Number of Interfaces per Component (ANIC): The ratio of the total number of interfaces to the total number of components.
9) Average Number of Connectors per Component (ANCC): the ratio of the total number of connectors to the total number of components.

3 The Experiment

It is widely accepted that software metrics are useless unless they can be of some practical use. Metrics can be of practical use for users, developers, managers, and team leaders in making predictions about some process features (e.g. cost/effort estimation, resources, etc.). Also metrics can be used to make predictions about the potential behavior of the system (Performance, reliability, efficiency, maintainability, etc..). Metrics help developers and managers detect the more complex components early at the design stage and make decisions to redesign these components.

In our validation we investigated the prediction models for both Function Points (FP) and development effort. Metrics data was collected from 30 projects which were designed for component oriented software development using COSEML [4] (Pronounced Cozumel). Most of the project topics related to either administrative management information systems or web-based applications. Projects were developed in teams of two or three graduate students in the department of Computer Engineering in METU. Projects vary in size (the largest project is estimated as 510 FP and the smallest project, 28 FP). Metrics have been collected by developers themselves. All developers were exposed to a short tutorial where metrics definitions and metrics collection approach have been explained.

Metrics collection forms were distributed to developers in order to facilitate the process of recording metrics data. Besides metrics values, developers provided records of:

- Function Points (FP).
- Development Effort (Person-Hours).

Regression analyses were performed using data from 25 projects while the remaining 5 projects were used for validating the prediction model. The nine variables (metrics) were fed in a forward addition approach. A variable qualified to enter the prediction model only if its coefficients corresponding p-value was less than 0.1. The obtained prediction model is defined as follows:

$$\begin{array}{l} FP = 0.8 * (TNC) + 4.3 * (TNIC) - 1.8 * (TNI) + 0.5 * (TNCO) + 12.7 \\ \quad \textit{(p = 0.01)} \quad \textit{(p = 0.03)} \quad \textit{(p = 0.06)} \quad \textit{(p = 0.1)} \quad \textit{(p = 0.02)} \end{array} \qquad (\mathbf{1})$$

Both statistical and practical significance of the model are observed. The statistical significance of the model is hinted by the high value of adjusted R^2 of 0.96 and the p-values of regressors' coefficients that are all within the 90% significance interval. The practical significance of the model is examined in two different ways:

1) comparing predicted FP values with those estimated from system requirements documents, an average error rate of %21 was obtained and after removing the outliers (highest 5 values of error rates), the average error rate dropped to %8, and
2) Five projects were not included in the model building process and used as a validation means for the model. The average error rate in these 5 projects was 10%.

In the second regression analysis we developed a prediction model for development effort (person-hours). Regressors are the same set of metrics used in the FP prediction model. Also in this model variables are fed in a forward addition approach. A confidence interval of 85% is used. The obtained prediction model is defined as follows:

$$\begin{array}{l} \text{Dev. Effort} = 2.4*(TNC) + 26.9*(TNIC) - 6.2* (TNI) + 9.8 *(ANCC) \\ \quad \textit{(p = 0.05)} \quad \textit{(p = 0.001)} \quad \textit{(p = 0.15)} \quad \textit{(p = 0.04)} \end{array} \qquad (\mathbf{2})$$

The model is practically significant due to the following reasons:

1) An average error rate of 17% was initially obtained. When removing outliers (highest 5 values of error rates) average error rate dropped to 9%.
2) An average error rate of 9% is achieved using the 5 projects which were not used in the regression model building process.

The complete lists of the original values and predicted values for both FP counts and development effort are presented in Table 1. The predicted values of FP and development effort are obtained using the formulas (1) and (2). Percentages of errors are estimated as shown in formula (3).

$$\text{Error Rate} = (\text{Actual value} – \text{Predicted value})/\ \text{Actual value} \qquad \textbf{(3)}$$

Table 1. Actual values and predicted values with the corresponding error rates for both FPs and Development effort

	FP Estimation Results			**Development Effort Estimation results**		
Project	**Actual value (FP)**	**Predicted value (FP)**	**Error %**	**Actual value (Person-hours)**	**Predicted value (Person- hours)**	**Error %**
1	142	128.06	9.82	480	481.43	-0.30
2	146	130.54	10.59	559	485.36	13.17
3	108	96.77	10.40	365	380.23	-4.17
4	85	87.45	-2.88	287	324.86	-13.19
5	66	62.16	5.82	253	239.89	5.18
6	113	82.83	26.70	391	315.89	19.21
7	65	54.29	16.48	225	203.46	9.57
8	44	56.22	-27.77	152	213.58	-40.52
9	28	35.97	-28.47	133	141.81	-6.62
10	66	62.34	5.55	223	234.01	-4.94
11	76	102.55	-34.94	204	375.89	-84.26
12	58	55.54	4.24	220	205.22	6.72
13	95	79.24	16.59	364	306.38	15.83
14	66	94.15	-42.65	235	362.87	-54.41
15	110	99.96	9.13	421	377.40	10.36
16	72	72.51	-0.71	349	268.12	23.17
17	102	107.47	-5.36	495	396.68	19.86
18	70	66.86	4.48	268	246.56	8.00
19	44	51.87	-17.89	167	196.25	-17.52
20	38	54.79	-44.20	145	209.36	-44.39
21	69	71.03	-2.94	246	263.97	-7.30
22	93	95.15	-2.31	356	371.32	-4.30
23	60	61.19	-1.98	230	227.85	0.93
24	48	51.32	-6.91	184	197.23	-7.19
25	510	513.76	-0.74	1953	1948.67	0.22

The curve fitting plots of the regression models are shown in the figures Fig. 1 and Fig. 2.

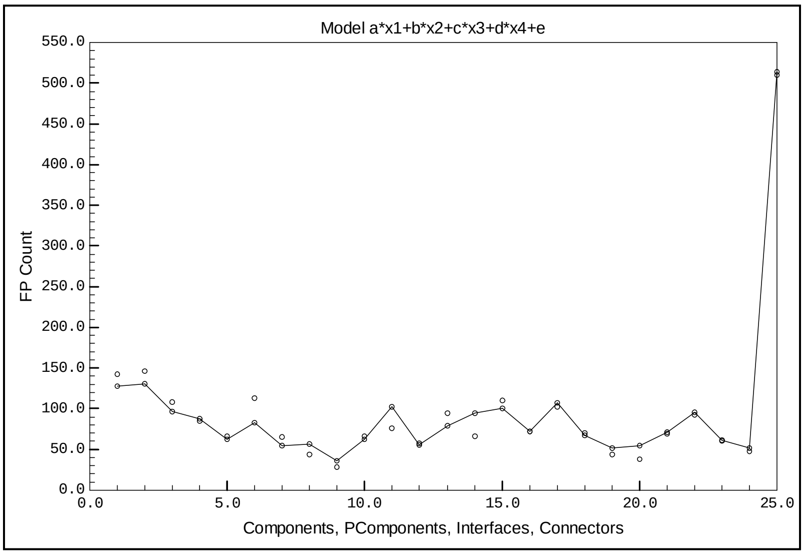


Fig. 1. Total FP Regression Model Plot

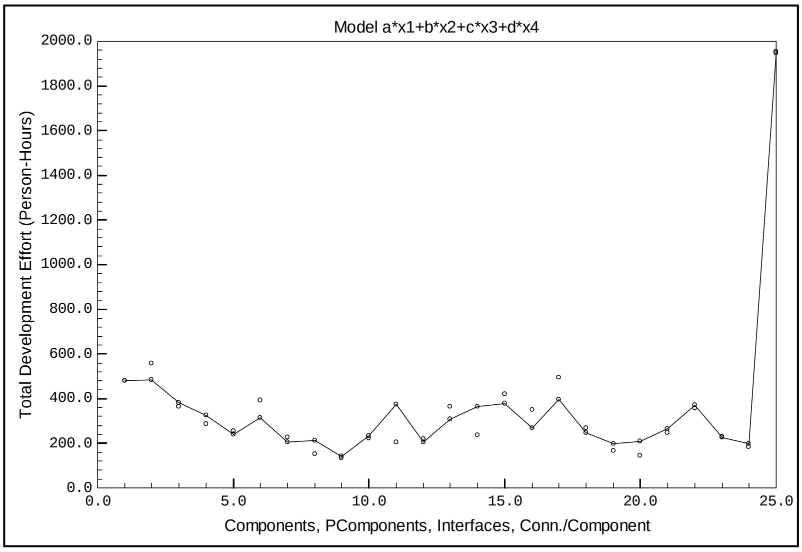


Fig. 2. Development Effort Regression Model Plot

4 Model Validation and Validation Threats

The obtained regression models have been further validated using data obtained from 5 projects. These 5 projects are also developed in similar environments to those used while building the regression models. These projects were selected randomly among the 30 projects included in the study. The error rates between actual and predicted values were in most cases below 10%.

In general, it can be clear that the following factors support the validity of the models:

1) Error rates for the prediction model are within the acceptable range of estimates for similar works for FP and development effort estimations.
2) Projects data were collected for three years from three different groups of students per year. This fact helped in considering projects prepared in diverse environments.
3) Developers voluntarily provided their data after the aims of the study were described to them. This suggests that developers were also interested in the study and eager to see CO as a mature methodology. This fact supports the trustability and hence, dependability of the data of these projects.

Although the results obtained from the validation are of great practical significance we still believe that further validation must take place using data from projects developed by the industry. The absence of industry practices in our validation is mainly due to two reasons:

1) CO is a new and emerging paradigm. No standard single CO development approach is used. In our research we followed the approach which was described in [4].
2) The lack of component oriented industry practices which are available for researcher experimentation.

5 Conclusion

The metrics have been evaluated using Tian and Zelkowitz axioms [8]. The experimental results show that structural complexity metrics can be of great managerial and practical use. Obtained results demonstrated slight deviations from the initial expectations: the coefficient sign of TNOI turned out negative in both models where the opposite might be expected through intuition. A possible explanation is that a negative TNOI is trying to balance the increase that is due to the TNIC coefficient.

Some product aspects which are intuitively believed to be related to some development effort could not be added to the regression models within the adjusted level of significance. The main notice-worthy among these are the DCT and TNM metrics.

Data to be collected from more projects in diverse environments, especially from the real industry can be used for further tuning of the models. Prediction models of reliability and efficiency for component oriented systems will be an important extension to this research.

References

1. Albrecht, A.J., Gaffney, J.E.: Software Function, Source Lines of Code, and Development Effort Prediction: A Software Science Validation. IEEE Transactions on Software Engineering , vol. SE-9, (1983) 639-648
2. Basili, V. R., Boehm, B.: COTS-Based Systems Top 10 List. IEEE Computer, vol. 34, No. 5, (2001) 91-93
3. Clements, P. C.: From Subroutines to Subsystems: Component-Based Software Development. American Programmer, vol. 8, no. 11 (1995)
4. Dogru, A. H., Tanik, M.: A Process Model for Component Oriented Software Engineering. IEEE Software, March/April, (2003) 34-41
5. Gorla, N., Ramakrishnan. R.: Effect of Software Structure Attributes on Software Development Productivity, Journal of Systems and Software, vol. 36, (1997) 191-199
6. Sedigh-Ali, S., Ghafoor, A., Paul, R. A.: Software Engineering Metrics for COTS-Based Systems. IEEE Computer, vol. 34, No. 6, (2001) 44-50
7. Szyperski, C., Gruntz, D., and Murer, S.: Component Software - Beyond Object-Oriented Programming. 2nd Ed. Addison-Wesley / ACM Press. (2002)
8. Tian, J., Zelkowitz, M. V.: Complexity Measure Evaluation and Selection. IEEE Transactions on Software Engineering, vol. 21, No. 8, (1995) 641-650
9. Vitharana, P., Zahedi, F. M., Jain, H.: Design Retrieval and Assembly in Component-Based Software Development. Communications of the ACM, vol. 46, No. 11, (2003) 97-102
10. Pressman, R. S.: Software Engineering: A Practitioner’s Approach. 6th ed., McGraw-Hill, (2005)
11. Salman, N.: Extending Object Oriented Metrics to Components: Proc. of the Sixth Biennial World Conference on Integrated Design and Process Technology, Pasadena, CA. (2002)
12. Salman, N. and Dogru, A. H.: Design Effort Estimation Using Complexity Metrics. Transactions of SDPS: Journal of Integrated Design and Process Science, vol. 8, No.3 (2004)
13. Chidamber, S. R., Kemerer, C. F.: Towards a Metrics Suite for Object-Oriented Design. Proc. Conf. Object-oriented Prog. Systems, Languages, and Applications (OOPSLA’91), vol. 26, No. 11, (1991) 197-211
14. Chidamber, S. R., Kemerer, C. F.: A Metrics Suite for Object-Oriented Design. IEEE Transactions on Software Engineering, vol. 20, No. 6, (1994) 476-493

Function Points Usage in Contracts – Considerations and Guidelines

Luca Santillo[1]

[1] Independent Consultant, GUFPI-ISMA Board Member,
Via A. Pollastrini, 7, 00062 Bracciano (RM), Italy
luca.santillo@gmail.com

Abstract. Since function point-like metrics cannot measure quality or technical aspects, the amount of Function Points is not sufficient to estimate a software project true cost. For long-term contracts, the average "cost per Function Point" could be extremely under- or over-estimated – from one project to another. Unfortunately, the seek for easiness and speed has privileged the spreading of fast project pricing mechanism – e.g fixed price per Function Point. A group of experts from the Italian Software Metrics Association issued a set of guidelines for customers and developers to agree upon the best usage of Function Points as a variable in negotiation and auditing of software development. The main factors affecting productivity are traced, as software reuse, change requests and quality, while trying to keep easiness and application flexibility.

Keywords: function point, contract, software, price, cost, guidelines.

1 Introduction

Function Points have been spreading in the software field since late 70's as an effective measure for functional software size [1]. Since such measure is not linked to any quality or technical aspect, it should be clear to practitioners that the only amount of Function Points for a software project is not sufficient to evaluate the right effort, duration and finally cost for the given project. For long-term contracts, this means that an average cost (price) per Function Point can be found to be extremely under- or over-estimated with respect to the real values of software development efforts, from project to project. Unfortunately, the seek for easiness and speed in software measurement and supply has privileged the spreading of fast pricing mechanism – the simplest and most dangerous being the fixed price per Function Point – in software development and enhancement projects.

In June 2006, GUFPI-ISMA (Gruppo Utenti Function Point Italia – Italian Software Metrics Association) issued a public domain technical report [2] comprising a set of guidelines for customers and developers to be able to agree upon the best usage of software measures, namely Function Points, as a variable of contractual negotiation and supplying auditing for ad hoc software development. A large number of practitioners – from both the customers' and the developers' side in Italy – and subject matter experts (including the author) did provide the national ICT community with a up-to-date analysis of do's and dont's, supply classes and adjustment aspects for usage of function point-like metrics in contracts. The main factors affecting productivity are traced in the guidelines, as software reuse, change requests and quality. Still, an attempt has being made by the guidelines' authors to provide easiness of use and flexibility in the application of the proposed approaches.

Following over ten years of Function Point usage in large-agreement environments in Italy, this first version of the GUFPI-ISMA guidelines summarizes the experience of practitioners and organizations in the field. This work aims to outline the guidelines' contents and concepts for the international audience, while stimulating the discussion for further improvements of the documented approaches.

2 Guidelines' Contents

Besides a proper introduction, where the main purpose, scope and involved roles are presented in detail with a domain-specific glossary of terms, the guidelines by GUFPI-ISMA consist of a main section, entitled "Function Points in Contracts for Ad Hoc (Custom) Software", and a series of appendices for in-depth presentation of specific subjects and hints for practical application. The main section of the guidelines tackles four areas:

- **Items to consider for a Function Point-based contract**
  - Supply classes
  - Lifecycle models and functional measurement moments
  - Supply scope
  - Software requirements types and detail levels
  - Change requests in the software lifecycle (incl. interrupted projects),
  - Pricing mechanism for interrupted projects
  - Price impacting factors
  - Base contractual schemes
- **Measures management in a custom software supply contract**
  - Call for tenders
  - Software requirement analysis
  - Software product implementation and final inspection
  - Software assets size update
  - Final definition of contractual pricing value
- **Some application models for described items**
  - Fixed price model for specified products/solutions
  - Variable price model for not-fully specified products/solutions
  - Variable price model for products/solutions to be specified
- **Pricing mechanism for custom software products supply**
  - Fair price definition
  - Quality of external benchmarking data
  - From effort to cost to price
  - The issue of moving average value

Appendices are provided about the following main topics:

- **Software functional size estimation methods**

  - Extrapolated measures
  - Sampled measures
  - Average complexity method
  - KISS method
  - Early & Quick Function Point
- **Examples of software process models and selection criteria**
- **Examples of effort percentages for software process models phases**
- **Software contractual value impacting factors**
  - Productivity adjustment factors
  - Contractual Functional Measure (MFC)
  - Reuse impact on contractual value
  - Replication impact on contractual value
  - Change request impact on contractual value
  - Quality measurement

The latest items comprise the most innovative concepts provided by the guidelines for a practical application of the concepts outlined in the main section, with respect to the current status of contractual usage of functional size, at least at the national level. Since describing every item in the guidelines is beyond the scope of this work, next sections addresses some of the concepts introduced or clarified by the guidelines.

3 Guidelines' Concepts and Clarifications (excerpt)

Supply classes. It is clarified that, in their current version, only the supply classes of "ad hoc (custom) software development" and "ad hoc (custom) software enhancement" are addressed by the guidelines. In this context, several types of maintenance activities are described, to help practitioners distinguish the proper domain of application. Hints or explanations are provided on excluded types, as "non functional enhancement" or "corrective maintenance".

Lifecycle models and functional measurement moments. Common models are included and described, as waterfall (with/without prototypes), incremental, iterative. So-called agile or extreme models are excluded in the current version, since they seem hard to apply in a contractual framework at the moment. For each included model, a table is provided describing when functional measurement can/should be applied within the given model process, with a rationale and estimated accuracy. Table 1 reports the example for the waterfall lifecycle model.

Software requirements types and detail levels. Following ISO classification, requirements are classified as *functional*, *technical* or *quality* requirements. On detail levels, it is clarified that the method applied to obtain the size must be explicated, since in certain stages for multi-year contracts no standard methods can actually be applied to high level project descriptions. Some estimation methods or techniques are provided in a related appendix (see section 4 for an excerpt).

Table 1. Determining Function Point measures in the waterfall lifecycle model.

When	How	Why	Accuracy (error)
Feasibility Study Completed (System User Requirements)	Functional size estimation methods	Provide essential information for decision making, feasibility test and project planning	Likely ±20-20%
Logical Design Completed (Architectural Design)	Standard methods (FP)	Provide the baseline for any further variation, change request or scope creep	Often <10%
On demand, along the lifecycle	Estimation or standard methods	Provide support to change request measurement and management	Depending on detail level
At completion	Standard methods (FP)	Provide contractual check of requested vs. delivered	Often <5%

Change requests. Since change requests can cause relevant delays or even failure of the project, this topic is described and addressed in detail by the guidelines, in two steps. For “interrupted” projects, a formula is provided for pricing approach to functionality that has been already analysed, designed, and or implemented when the project is stopped, as follows:

$$\text{Reduced Price} = \text{FP} \times (\text{Unit Price} \times \text{Performed phases cost percentage})\ , \qquad (\mathbf{2})$$

where the parenthesis stress the fact that conceptually the price adjustment is applied to the unit price, based on the percentage of the work effort performed before the project has been stopped, rather than to the Function Point amount.

A further step for more accurate determination of change request impact is provided in a related appendix (see section 4 for an excerpt).

Price impacting factors. A sample of impacting factors, suggested by several research and industrial models, is provided. A first classification of the factors is suggested based on product, process, technology, and staff domains. Clarification is explicated on the fact that such list is not exhaustive, and that different models can denote one factor with different terminology, or on the opposite the same terminology in different models could refer to different factors from a conceptual point of view. The listed factors are then mapped on a Relevance/Measurability diagram, with scales from low to high impact relevance or strength on pricing values and ease or capability of measurement. Such diagram can help project/program managers to identify the most significant factors to measure, manage and (possibly) control. Figure 1 reports the cited diagram (a caption describing the low/average/high levels for relevance and measurability is provided in the guidelines – hereby not reported).

		Measurability		
Impact Relevance		Low	Average	High
	High	Analysts capability **Programmers capability** Technical reuse …	**Change requests (requirements volatility)** Complexity …	Supply class **Functional reuse** Software replication **Required reliability** …
	Average	**Process maturity** Analysis/Design methods **Available utilities** Programming methods …	Resources availability **Architecture** System type **Required documentation** …	**System integration** System interfaces …
	Low	Project Manager capability **Team cohesion** …	**System performance** …	Database volume …

Fig. 1. Impacting factors map, with respect to the capability to measure them and the relative impact they can have on the pricing mechanism (excerpt sample).

Base contractual schemes. Four models are identified and briefly described for contractual frameworks:

- Body Rental
- Time & Material
- Measure-based
- Product-based

For each type, a definition, risk elements, pricing definition mechanism, change request treatment and warranty possibilities are provided.

Body Rental and Time & Material types are not further developed or addressed in the guidelines. For the latter types, application models are suggested in a specific subsection of the guidelines (see section 4.1 for a brief excerpt).

Measures management. Another aspect discussed in-depth by the guidelines is the estimation or measurement activities related to the generic steps that can be identified in any contractual framework development process. For instance, main stages of any "contract process" are listed as:

- Pre-contract start-up (needs definition and call for tenders)

- Contract execution (several phases, typically related to the software process itself)
- Contract closure (assets update and economical check-out)

For each stage, estimation or exact measurement of the functional size of software products are proposed where applicable.

4 Guidelines' Suggested Approaches (excerpt)

This section reports on some of the approaches suggested by the guidelines to address the main aspects and concepts previously mentioned.

4.1 Some (contract) application models for described items

The models considered are:

- Fixed price model for specified products/solutions
- Variable price model for not-fully specified products/solutions
- Variable price model for products/solutions to be specified.

For each model, the following aspects are discussed in the guidelines, obviously with different results from one model to another:

- general consideration
- initial definition of prices
- modalities for software metrics auditing
- modalities for change request management
- modalities for definition review of prices
- possible extensions to the main model.

The most developed model is the third one (variable price for products/solutions to be specified), of particular interests for mid to long-term contracts, where the initial setup of the contract is simply not able to define in advance all the projects and – for each project – the exact requirements that will be specified during the contract execution (usually, spanning over a time period much longer than one year).

In the mentioned model, the initial definition of prices section suggests to consider a derived (indirect) measure, denoted as Contractual Functional Measure (MFC, Misura Funzionale Contrattuale) as the product measure to relate to the unit price in the contract. Two variations are possible to combine the relevant impacting factors:

- project class-based approach
- adjustment factors multiplication-based approach

In the project class-based approach, the contract should define a (reduced) set of project types (classes) based on the most common and realistic combinations of impacting factors, and a corresponding set of unit prices per function point per each project class.

In the adjustment factors multiplication-based approach, in a similar way derived by COCOMO-like models [3], the contract should define a set of impacting factors and a scale of adjustment values per each factor over the pricing value.

(In reality, it is possible to prove that the two approaches are equivalent one to each other, under general conditions.)

Figure 2 and Figure 3 show the overall models for initial price definition for a given project under the discussed contract model (variable price fro products to be specified).

Besides injection of, respectively, project class values and factor adjustment values, both models include reuse and replication concepts to derive the Contractual Functional Measure. Reuse and replication are further described in corresponding appendices in the guidelines; further aspects, as intrinsic complexity, could be addresses in future versions of the guidelines, provided that double consideration is avoided – any factor explicitly considered in the MFC measure must not be also considered to derive a project class or as an pricing adjustment actor.

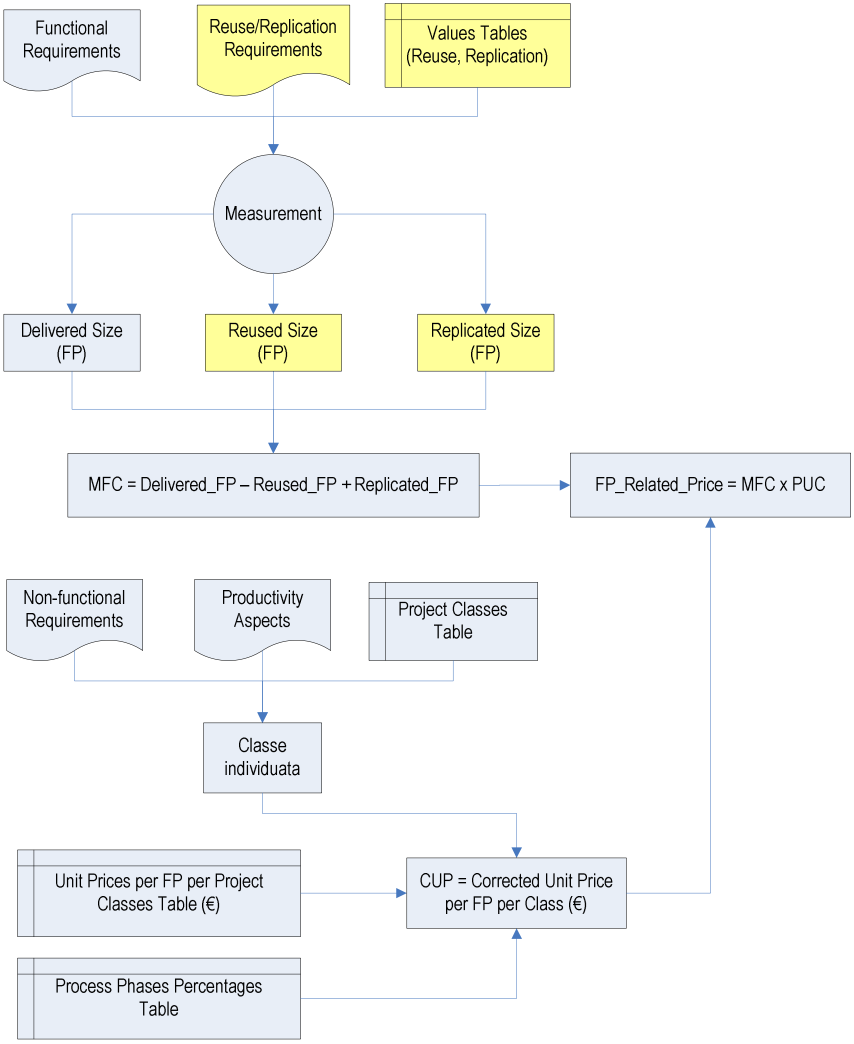

Functional Requirements
Reuse/Replication Requirements
Values Tables (Reuse, Replication)
Measurement
Delivered Size (FP)
Reused Size (FP)
Replicated Size (FP)
MFC = Delivered_FP – Reused_FP + Replicated_FP
FP_Related_Price = MFC x PUC
Non-functional Requirements
Productivity Aspects
Project Classes Table
Classe individuata
Unit Prices per FP per Project Classes Table (€)
CUP = Corrected Unit Price per FP per Class (€)
Process Phases Percentages Table

Fig. 2. Contractual Functional Measure (MFC) diagram, based on project classes.

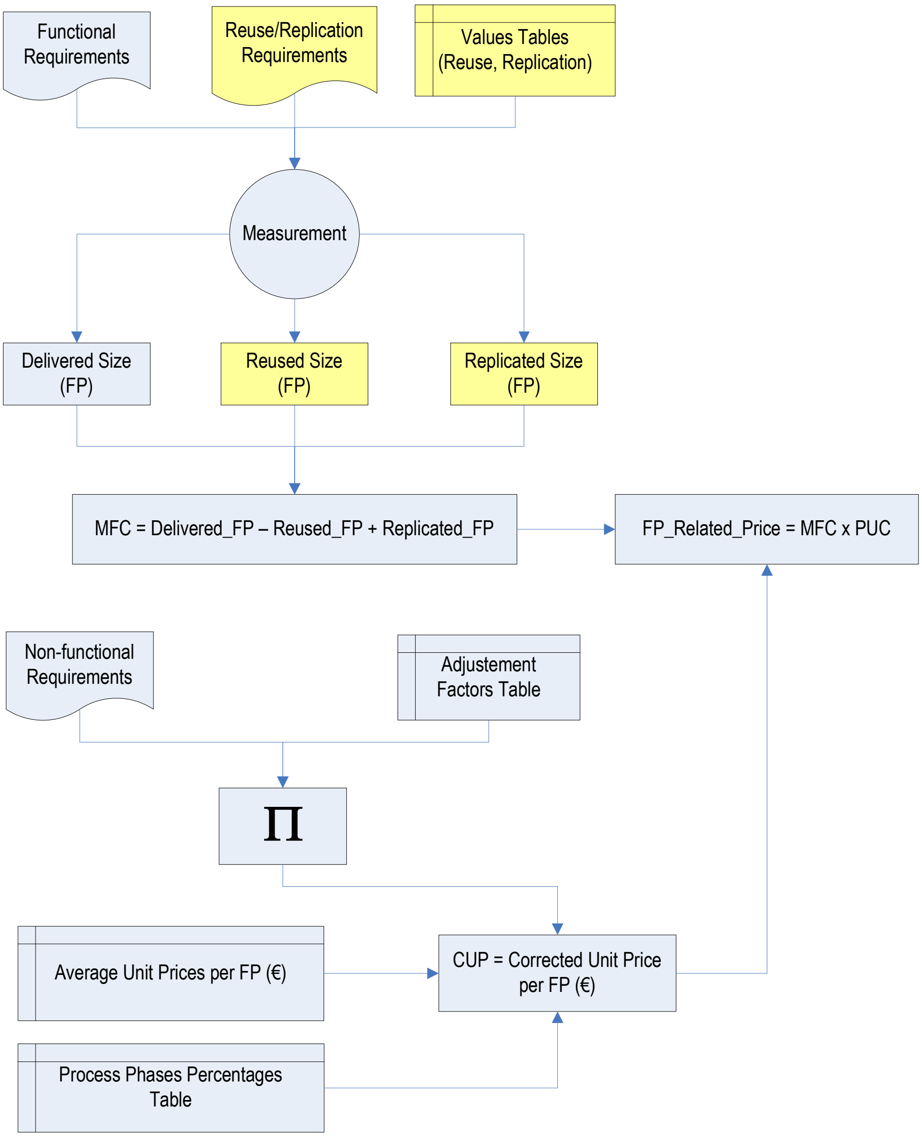

Functional Requirements
Reuse/Replication Requirements
Values Tables (Reuse, Replication)
Measurement
Delivered Size (FP)
Reused Size (FP)
Replicated Size (FP)
MFC = Delivered_FP – Reused_FP + Replicated_FP
FP_Related_Price = MFC x PUC
Non-functional Requirements
Adjustement Factors Table
Π
Average Unit Prices per FP (€)
CUP = Corrected Unit Price per FP (€)
Process Phases Percentages Table

Fig. 3. Contractual Functional Measure (MFC) diagram, based on adjustment factors. The upper portion in as in Figure 2. **Π** stands for the "product of *n* given factors values".

4.2 Pricing mechanism for custom software products supply

A discussion is provided in the guidelines about the issue of a "fair price definition", where practitioners need to fix an initial price for a given project class, or the average price per Function Point. Three main questions are discussed, namely:

- the quality of external benchmarking data
- the translation from effort to cost to price
- the issue of moving average value

Some hints are provided in the guidelines on such questions.

4.3 Software functional size estimation methods (Appendix)

Several techniques are reported, to obtain an early approximation of functional size, when needed. A description of six levels accuracy definitions for size measures is also provided, based on ISBSG publications [4]. A set of references is set, pointing to detailed descriptions of the suggested techniques (e.g. [5] for Early & Quick FP).

4.4 Examples of software process models and selection criteria (Appendix)

A list of phases, standard deliverables, and phase closure criteria is provided, for five development models (traditional/full, traditional/reduced, unique phase, knowledge-based, object oriented. Such lists can serve as a template for real contract definition.

4.5 Examples of effort percentages for software process phases (Appendix)

Based on public sources spanning over the last five years, some effort percentages values are reported, to help customers and developers negotiate their own values in real contract definition.

4.6 Software contractual value impacting factors (Appendix)

Several aspects are discussed in-depth in this part of the guidelines. For instance, a list of productivity factors, together with a possible value scale is proposed deriving from COCOMO models [3]. Contractual Functional Measure, Reuse and Replication impact factors are further discussed, since they play a primary role in the models previously reported (e.g. refer to Fig. 2 and 3).

On Change Request, a specific section provides a method and a formula to determine the size of a change request in terms of what is added, changed, and deleted with respect to the initial baseline measure. The change request measurement can serve as both a way to correct the Contractual Functional Size, if approved, in the final pricing mechanism of the given contract, or as an indicator of the overall requirements volatility, to explain delays or rework efforts, and to derive a specific productivity factor.

Finally, an initial selection of quality aspects is provided, as a trace for practitioners to consider, and for further evolution of the guidelines on such aspects.

5 Conclusions

As pointed out by the previous sections and excerpts, several distinct aspects of the usage of Function Points in contractual frameworks are discussed and proposed for suggested solutions in the current first version of GUFPI-ISMA guidelines. The main challenge of such work is to merge such aspects in an organic and systematic view, while keeping ease of use and flexibility for practitioners to manage different types of contracts in a consistent way. Undoubtedly, a long path is to be covered to reach such goal. Nonetheless, thanks to the contributions of so many subjects among the authors and reviewers of the guidelines, this first global attempt at national level for Italy helps in making explicit what to consider as base elements for a proper usage of functional size measures in software contracts, and what to avoid (first release "do's and don't's"). A first positive, practical result of this work should come from the inclusion of many concepts and suggested approaches into another set of guidelines on product and services quality in public administration by CNIPA (the National Centre for Information Technology in Public Administration in Italy) [6], which is under study at the moment this paper is issued. Such study, together with industry tests and further experts contributions, will lead to a an improved version of the GUFPI-ISMA guidelines in the future. Finally, a comparison with other national bodies frameworks on the subject of contractual usage of software metrics would result in a enriched and more consistent set of globally accepted practices.

Acknowledgments. The author wishes to thank all GUFPI-ISMA members, researchers and coordinators who helped in providing his and others' contributions, and in merging them in a consistent way in the guidelines. A complete list of authors and reviewers of the guidelines hereby briefly discussed is reported in [2].

References

1. IFPUG: Function Point Counting Practices Manual, Release 4.2.1, International Function Point Users Group (2006)
2. GUFPI-ISMA: LGC-FP – Linee Guida per l'Uso Contrattuale dei Function Point, Versione 1.0, Documento Tecnico 2006/01, Gruppo Utenti Function Point Italia – Italian Software Metrics Association, Italy (2006)
3. Boehm, B.W. et al.: Software Cost Estimation with COCOMO II, Prentice-Hall (2000)
4. ISBSG: Practical Project Estimation, 2nd edition", International Software Benchmarking Standards Group, Australia (2005)
5. Santillo, L., Conte, M., Meli, R.: Early & Quick Function Point – misurare di più con meno, in: GUFPI-ISMA (ed.): Metriche del software, Esperienze e ricerche, pp. 196-210, Franco Angeli, Italy (2006)

6. CNIPA: Linee guida sulla qualità dei beni e dei servizi ICT per la definizione e il governo dei contratti della Pubblica Amministrazione, National Centre for Information Technology in Public Administration, Italy (2004)

Strategies and Appropriateness of Software Measurement Frameworks

Reiner R. Dumke, René Braungarten[1], Martin Kunz, Andreas Schmietendorf[2], Cornelius Wille[3]

University of Magdeburg, Faculty of Computer Science
{dumke,makunz}@ivs.cs.uni-magdeburg.de
[1]Bosch Rexroth, Lohr, Germany, braungar@ivs.cs.uni-magdeburg.de
[2]Berlin School of Economics, Berlin, Germany, schmiete@fhw-berlin.de
[3]University of Applied Science, Bingen, Germany, wille@fh-bingen.de

Abstract. This paper describes the general background of software measurement frameworks, the current situation in the context of existing standards and methodologies and the future aspects considering the Web-based technologies in the form of e-measurement. A first part about the formal background of software measurement should help to understand the appropriateness of measurement approaches and methods. The description of the software measurement frameworks will be started with a general framework explanation and the characterization of declarative versus operational measurement frameworks. A framework is presented *as CAME approach* which combines the declarative and operational characteristics. The new kinds and possibilities of Web-based software measurement (as *e-measurement*) are described in a final section including first examples and then components of Web-based measurement infrastructures.

1 Introduction

Considering the different kinds of measurement activities we can characterize the dependencies in the following simplified manner [3].

We assume that these measurement activities are well known and consider the different types of measurement frameworks in following (see also [1], [5] and [9]). Measurement frameworks could be divided in general in two classes: the declarative and the operational measurement frameworks.

Declarative software measurement frameworks describe *what* we achieve based on the software measurement methods and activities. *Operational software measurement frameworks* characterize *how* we achieve the measurement goals and levels. In order to characterize these different kinds of measurement frameworks we will use the following general description of software measurement. The first figure (figure 2) shows the measurement granularities as a layer model in a declarative manner.

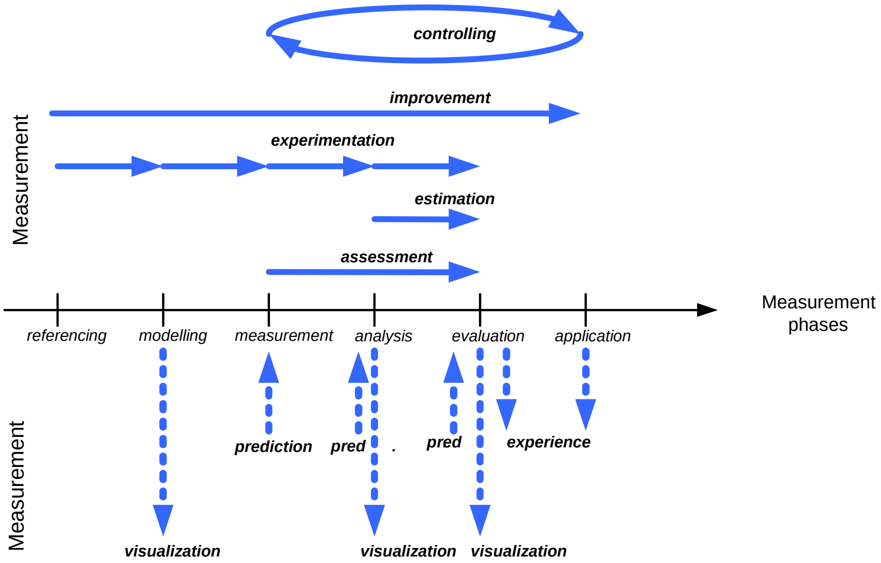


Fig. 1: Software measurement phases and methods

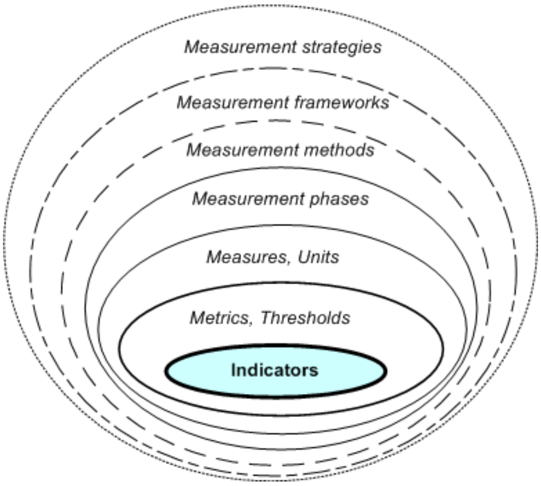


Fig 2: General aspects of software measurement

On the other hand, considering measurement operations *M* we will use the following schema of measurement processes and phases respectively.

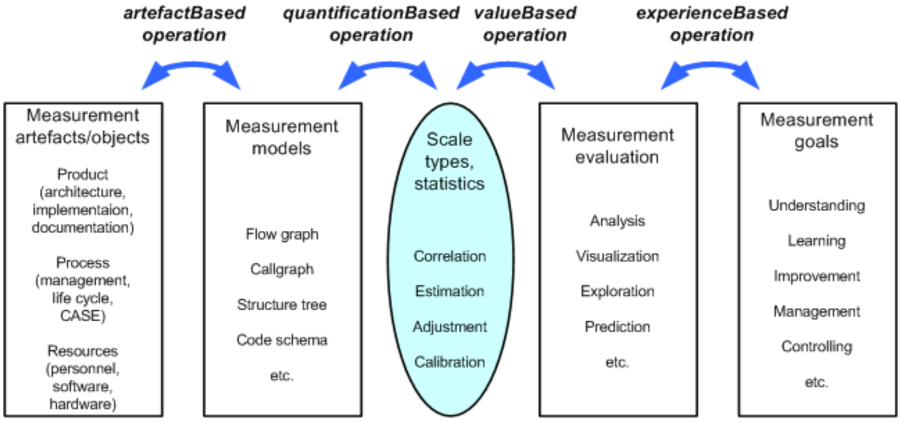


Fig. 3: Software measurement operations

2 Basic Characteristics of Software Measurement Frameworks

There are plenty of measurement frameworks (SPICE, Bootstrap, EFQM, ISO 19761 etc.) but we consider only two kinds of frameworks in this paper.

Declarative Measurement Frameworks

At first we will consider the ***Zuse Measurement Framework*** which includes the largest and profoundest investigations of existing metrics approaches [15]. In following we will only characterize some of the essential intentions of this measurement framework.

- The following kinds of *requirements for software measures* are defined by Zuse:

$$requirementsForMeasures = \{req_{general}, req_{rankingProperties}, req_{units}, req_{substitutionOperation}, req_{additiveRatioScale}, req_{non\text{-}additive}, req_{densityMeasures}, req_{wholeness}, req_{predictionModels}, req_{sizeMeasures}, req_{simpleMeasures}, req_{componentIndependence}\} \quad (2.1)$$

- The *types of measures* based on the measurement theory are the following

$$typesOfMeasures = \{TM\text{-}CNT, TM\text{-}ADD, TM\text{-}HYA, TM\text{-}HYAS, TM\text{-}DEN, TM\text{-}DENM, TM\text{-}NADDR, TM\text{-}BSA, TM\text{-}HYM, TM\text{-}ORD, TM\text{-}PC, TM\text{-}FB, TM\text{-}MIMA, TM\text{-}IND, TM\text{-}NGEXT\} \quad (2.2)$$

where means *TM-CNT* (counting on attribute), *TM-ADD* (additive measure), *TM-HYA* (hybrid measure by additivity), *TM-HYAS* (hybrid measure additive plus a constant), *TM-DEN* (density measures), *TM-DENM* (density measure with an additional condition), *TM-NADDR* (non-additive ratio scale), *TM-BSA* (two concatenation operations), *TM-HYM* (hybrid measure by multiplication), *TM-ORD* (purely ordinal measure), *TM-PC* (percentage measure), *TM-FB* (modified function of belief), *TM-MIMA* (minimum – maximum), *TM-IND* (independence conditions), *TM-NGEXT* (negative extension structure and non-additive ratio scale);

- The following kinds of measurement of *artefacts* are considered in detail in the Zuse framework:

  $A_{framework}= \{A_{ISO9000-3}, A_{functionPoints}, A_{COCOMO}, A_{maintainability}, A_{structureCharts}, A_{informationFlow}, A_{measuresOfBowles}, A_{coding}, A_{testing}, A_{cohesion}, A_{maintenance}, A_{documentQuality}, A_{OOmeasures}, A_{lifeCycle}\}$ (2.3)

- In the *scale analysis* the following operations about the components of the measured artefact are helpful:

  $M_{modelling} \in \{BALT, BSEQ, CINT, CUNI, DSEQ, HAGG, HGEN, LSEQ, RALT, RSEQ, SBSEQ\}$ (2.4)

  where means *BALT* (parallel concatenation), *BSEQ* (sequential concatenation), *CINT* (object-oriented concatenation), *CUNI* (object-oriented unification), *DSEQ* (charts sequential concatenation), *HAGG* (object-oriented hierarchical concatenation), *HGEN* (object-oriented generalizational concatenation), *LSEQ* (boards sequential concatenation), *RALT* (resistors parallel concatenation), *RSEQ* (resistors sequential concatenation), *SBSEQ* (source code sequential concatenation);

- The application of the *viewpoint principle* for using metrics. The viewpoint and the empirical system are identical. The term viewpoint is more intuitive for the user and well known from the daily life. (2.5)

- The *principles of metrics application* are defined by Zuse as following:

  1. *There is no best measure.* (2.6)
  2. *One number cannot express software quality.*
  3. *The properties of the used measure should be well known.*
  4. *One measure is not sufficient because software systems consist of many properties.*
  5. *A so-called software quality or maintainability index, combined by many measures, has to be considered carefully.*
  6. *It is a widely spread misunderstanding, that validated measures only are useful.*
  7. *It is important to know whether a measure assumes an extensive structure or not.*
  8. *A measure that is validated in one environment is not automatically a validated measure.*
  9. *A measure that has a good correlation to a validated measure is not automatically validated, either.*
  10. *The environment in every company for measurement is differently.*

11. *There does not exist an initial set of measures or even a final set of measures.*

The application of the software measurement framework by Zuse keeps the consideration of the measurement theoretical aspects for achieving really measures and their correct involvement in the software evaluation.

Now we will consider the ***ISO 15939 measurement standard*** that includes the phases of establishing and sustain measurement commitment, plan the measurement process, perform the measurement process, and evaluate measurement [7]. The following figure shows essential components of this standard.

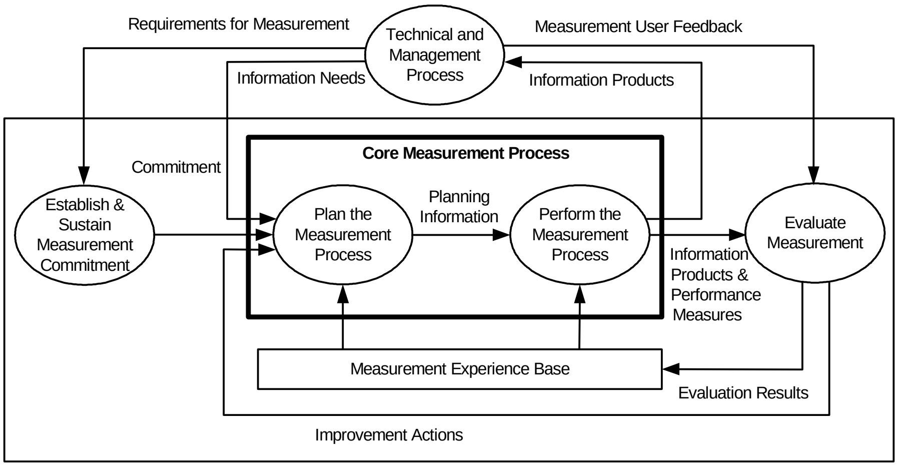

Requirements for Measurement
Technical and Management Process
Measurement User Feedback
Information Needs
Information Products
Core Measurement Process
Commitment
Establish & Sustain Measurement Commitment
Plan the Measurement Process
Planning Information
Perform the Measurement Process
Evaluate Measurement
Information Products & Performance Measures
Measurement Experience Base
Evaluation Results
Improvement Actions

Fig. 4: The ISO 15939 measurement standard

The basic tasks and artefacts/objects are:

establishingTasks = {acceptanceOfMeasurement, assignTheResources} 2.7)

establishingArtefacts = {managementCommitment, measurementRequirements, planForMeasurementResources, descriptionOfTheOrganisationalUnit}

planningTasks = {organisationCharacterization, needsIdentification, measuresSelection, evaluationProcedures, evaluationCriteria, measurementResourcing, measurementSupporting} (2.8)

planningArtefacts = {informationNeeds, descriptionOfTheOrganisationalUnit, candidateMeasures, selectedMeasures, measurementTasks, informationProducts, toolsDescriptions, trainingCourseMaterials} *performingTasks* = {*processIntegration, dataCollection, dataAnalysis, userCommunication*} (2.9)

performingArtefacts = {*measuredArtefacts, collectedData, storedData, informationProducts, measurementExperienceBase*}

evaluationTasks = {*measurementEvaluation, measurementImprovement*}*evaluationArtefacts* = {*evaluationCriteria, informationProducts, lessonLearned, measurementExperienceBase*} (2.10)

explorationTasks = {*measurementReporting, measurementExploration, measurementConclusion*}

explorationArtefact = *measurementExperienceBase*

controllingTasks = {*measurementFeedback, informationGeneration*} (2.11)

controllingArtefacts = {*informationProducts, informationNeeds*}

The figure 5 shows the general intentions of the described software measurement frameworks. Note that these frameworks are mainly described in a *declarative manner.*

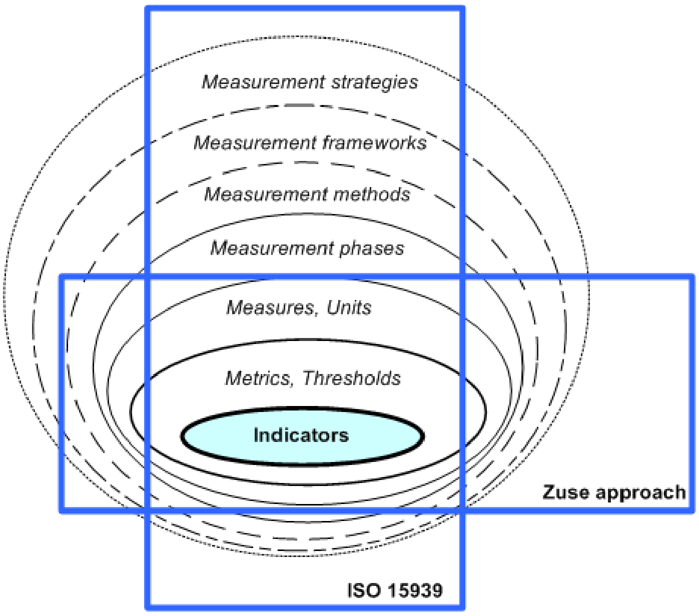


Fig. 5: Measurement characteristics of two declarative measurement frameworks

Operational Measurement Frameworks

We will consider the GQM and the Six Sigma approach as examples of operational measurement frameworks in following.

The ***Goal Question Metric (GQM) approach*** points out that the existence of explicitly stated goals is of the highest importance for improvement programs [12]. GQM presents a systematic approach for integrating goals to models of the software processes, products and quality perspectives of interest, based upon the specific needs of the project and the organization. In other words, this means that in order to improve a process you have to define measurement goals, which will be, after applying the GQM method, refined into questions and then into metrics that will supply all the necessary information for answering those questions. The GQM methodology contains four phases [6].

- The planning phase in which the project for measurement application is selected, defined, characterized and planned, resulting in a project plan. (2.12)
- The definition phase, during which the measurement program is defined (goal, questions, metrics and hypotheses are defined) and documented. (2.13)
- The data collection phase includes the actual data collection which takes place, resulting in collected data. (2.14)
- In the interpretation phase the collected data is processed with respect to defined metrics into measurement results, which provides answers to defined questions, after which goal attainment can be evaluated. (2.15)

The GQM method provides a measurement plan that deals with the particular set of problems and the set of rules for interpretation of the obtained data.

The ***Six Sigma approach*** has its origins in statistics using the symbol for standard deviation [13]. Is has also become a management philosophy that includes the need for fact-based decisions, customer focus, and teamwork. The main focus is related to error deviation with different levels of considerations (usually from 3σ to 6σ).Six Sigma is based on the *DMAIC model* which involves the cyclic phases as *define, measure, analyze, improve,* and *control.* Considering the measurement process involvements we can establish the following characteristics:

> The key steps within the *definition phase* are: define the problem, form a team, establish a project charter, develop a project plan, identify the customers, identify key outputs, identify and prioritize requirements, and document the current process. The key steps within the *measurement phase* are: determine what to measure, conduct the measurements, calculate the current sigma level, determine the process capability, and benchmark the process leaders. (2.16)
>
> The key steps within the *analysis phase* are: determine what caused the variation, brainstorm ideas for process improvements, determine which

improvements would have the greatest impact on meeting customer requirements, develop a proposed process map, and assess the risks associated with the revised process. (2.17)

The key steps within the *improvement phase* are: gain approval for the proposed changes, finalize the implementation plan, and implement the approved changes. (2.18)

The key steps within the *control phase* are: institutionalize the process improvements so that the changes are permanent and gains are sustained, develop and communicate metrics that continue to reinforce the value of the improvements, and establish mechanism for dealing with out-of-control situations. (2.19)

The following figure 6 shows the general intentions of the described software measurement frameworks in an *operational manner*.

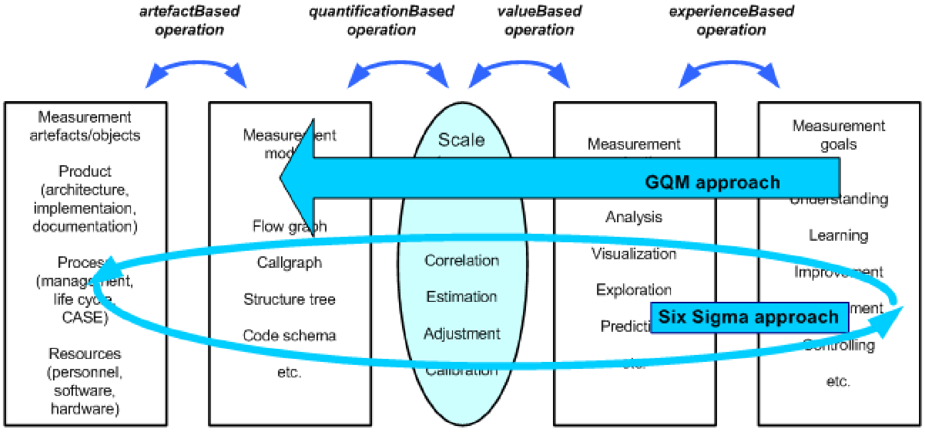


Fig. 6: Measurement characteristics of some operational measurement frameworks

Furthermore, there are a lot of formal approaches for software measurement including operational or framework aspects. But, we only will shown these approaches without any further discussion (see the details in [5]).

3 The CAME Approach

We use the acronym CAME in three meanings to meet the requirements of software measurement. The CAME aspects are defined in a layer model which consists of a kernel CAME tools, a general CAME strategy with an embedded CAME framework.

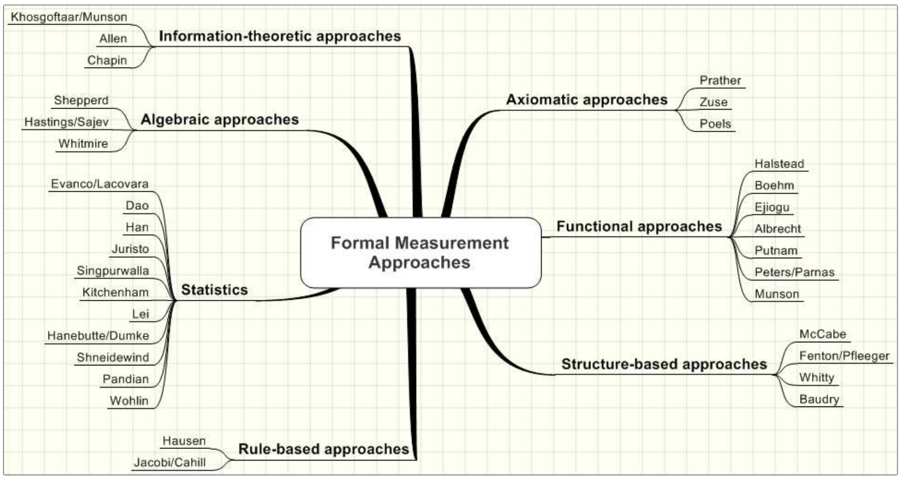


Fig. 7: Overview of formal software measurement approaches

The *CAME strategy* stands for [3]

- *Community:* the necessity of a group or a team that is motivated and has the knowledge of software measurement to install software metrics. In general, the members of these groups are organized in metrics communities such as our German Interest Group on Software Metrics (http://ivs.cs.uni-magdeburg.de/sw-eng/us/giak/).

- *Acceptance:* (is based on (2.6), (2.12)) the agreement of the (top) management to install a metrics program in the (IT) business area. This aspect is strong connected with the knowledge about required budgets and personnel resources.

- *Motivation:* (is based on (2.6), (2.16)) the production of measurement and evaluation results in a first metrics application which demonstrates the convincing benefits of the metrics application. This very important aspect can be achieved by the application of essential results in the (world-wide) practice which are easy to understand and should motivate the management. One of the problem of this aspect is the fact that the management wants to obtain *one single (quality) number* as a summary of all measured characteristics.

- *Engagement:* (is based on (2.14), (2.15)) the acceptance of spending effort to implement the software measurement as a permanent metrics system (with continued measurement, different statistical analysis, metrics set updates etc.). This aspect includes also the requirement to dedicate personnel resources such as measurement teams etc.

This is one of the reasons that this approach should consider the whole sides of declarative measurement frameworks simplified shown in the following figure.

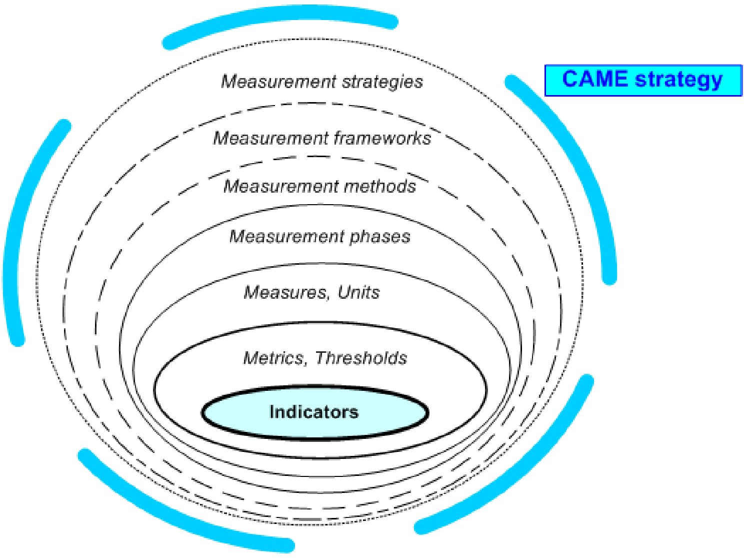


Fig. 8: The CAME strategy based declarative frameworks involvements

The *CAME framework* itself consists of the following four phases:

- *Choice:* the selection of metrics based on a special or general measurement view on the kind of measurement and the related measurement goals -- as *measurement areas*,

- *Adjustment:* the measurement characteristics of the metrics for the specific application field – as *metrics levels*,

- *Migration:* the semantic relations between the metrics along the whole life cycle and along the system architecture – as *characteristics of the measurement methods and their integration level* ,

- *Efficiency:* the automation level as construction of a tool-based measurement – as *implementation of the measurement methods* by *CAME tools (computer assisted measurement and evaluation tools).*

The following figure characterize the CAME intentions addressed to operational aspects of software measurement frameworks.

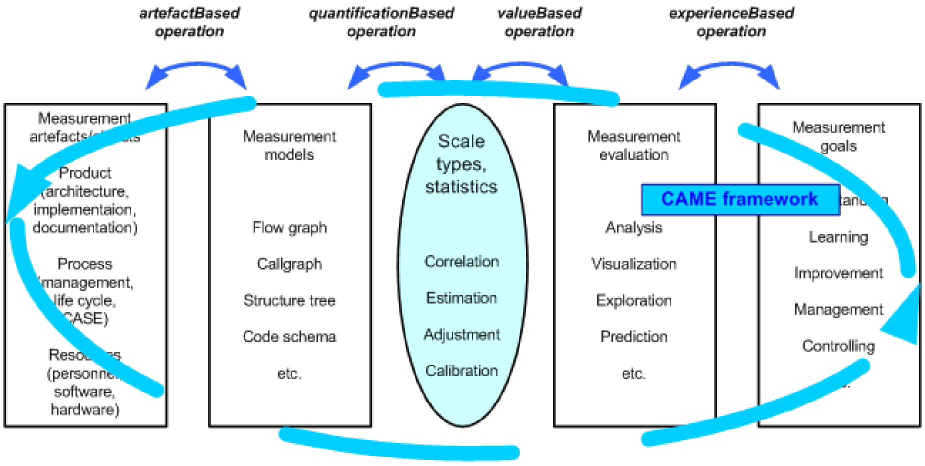


Fig. 9: The CAME framework based operational frameworks involvements

Note that the "operational cycle" seems to be like in the Six Sigma approach but considers the whole spectrum of the empirical aspects and not the defects only.

In order to fulfill the operational measurement aspects the CAME framework includes following principles and rules:

Choice characteristics:

Here we decide about the *coverage* of our measurement areas relating to the product, process and resources or - otherwise - which means that we decide which areas (components, artifacts) are "out of control". Dividing the measured artifacts in *design*, *description* and *working* we can define the following ***choice levels*** of measurement:

- *No measurement:* no kinds of measurement and evaluation are applied and used for the considered IT area
- *Aspect-oriented measurement*: only some characteristics, criteria or entities were considered
- *Capability-oriented measurement*: some sub areas or sub domains were considered in the IT area
- *Total measurement*: in the sense of full/total quantitative analysis and management of the IT area.

The following figure shows a compact presentation of the choice level from our point of view adapted and extended from Wille [14] considering the declarative part of the CAME framework.

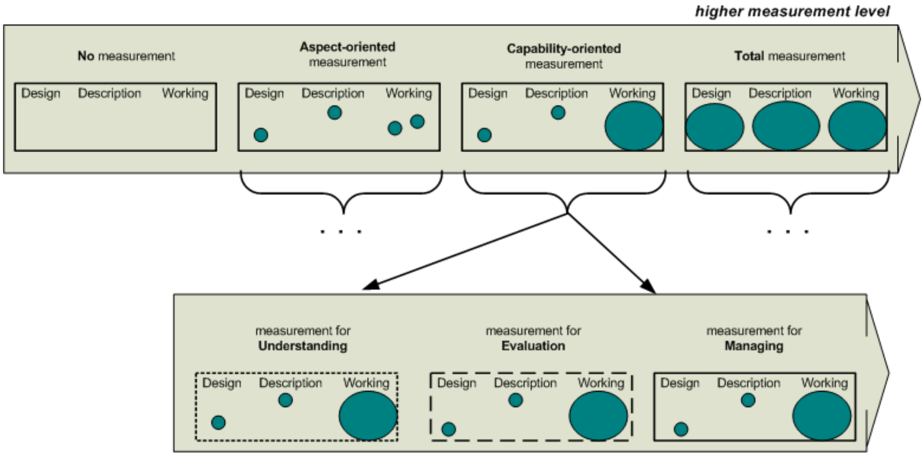

higher measurement level
No measurement
Aspect-oriented measurement
Capability-oriented measurement
Total measurement
Design
Description
Working
. . .
. . .
measurement for Understanding
measurement for Evaluation
measurement for Managing

Fig. 10: The CAME levels choosing/selecting the measurement areas

Choice principles: (are based on (1.1), (2.5), (2.6), (2.8), (2.13), (2.16))

In order to achieve a wide-range measurement we could use the following helpful activities in order to improve the current software measurement level ([5], [14]). In the case of *missing metrics* for the considered artefacts or measurement area it would helpful to use existing metrics from the analog paradigms or technologies.

Adjustment characteristics:

The quality of the chosen metrics was involved by their scale characteristics which mean that we can establish the situation of *qualitative measurement* (achieving a nominal or ordinal scale) or of *quantitative measurement* (achieving an interval or ratio scale). On the other hand, the used metrics characteristics like thresholds could be applied from another technology (as *approximation*) or from another product of the same technology (as *tuning*). Both characteristics determine the ***adjustment level***. The following figure combines these aspects of *adjustment levels* with the choice level defined above.

Adjustment principles: (are based on (1.1), (2.1), (2.2), (2.4), (2.10), (2.17))

In the case of *missing thresholds* empirical values could be taken from another experience like analogies, expertises or rules of thumb. The situation of *un-clarified scale characteristics* could be improved by the initial use of the lower level scale type such as qualitative measurement with nominal and ordinal scales.

Migration characteristics:

The migration step of the CAME framework involves the tasks of integration, aggregation and composition of the measurement phases based on measurement methods or frameworks in the IT processes of development and maintenance of software systems. In a simplified manner we could decide in measurement (including the modeling, measurement and evaluation) and *controlling* (including the measurement application and building a “measurement cycle” furthermore) as two kinds of the ***migration level*** considering the operational part of the CAME framework.

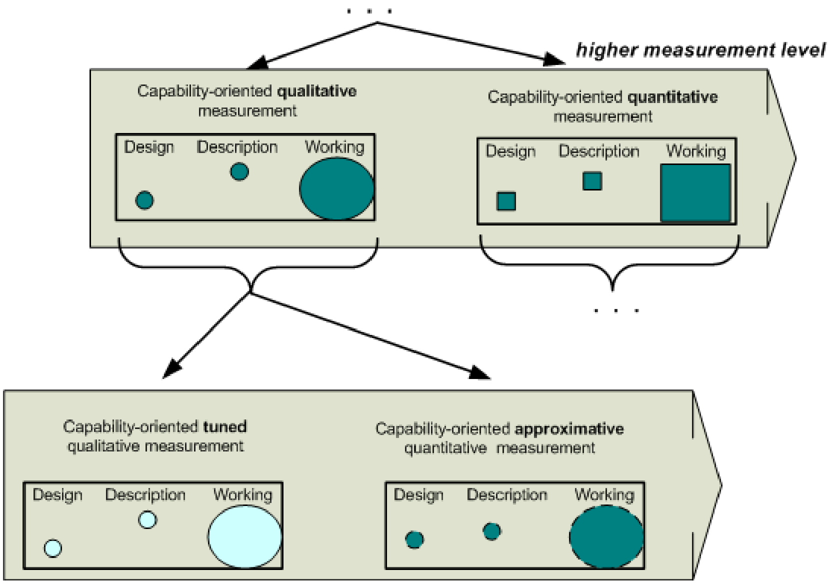

. . .
higher measurement level
Capability-oriented qualitative measurement
Design Description Working
Capability-oriented quantitative measurement
Design Description Working
. . .
Capability-oriented tuned qualitative measurement
Design Description Working
Capability-oriented approximative quantitative measurement
Design Description Working

Fig. 11: The CAME levels considering the scale types of chosen metrics

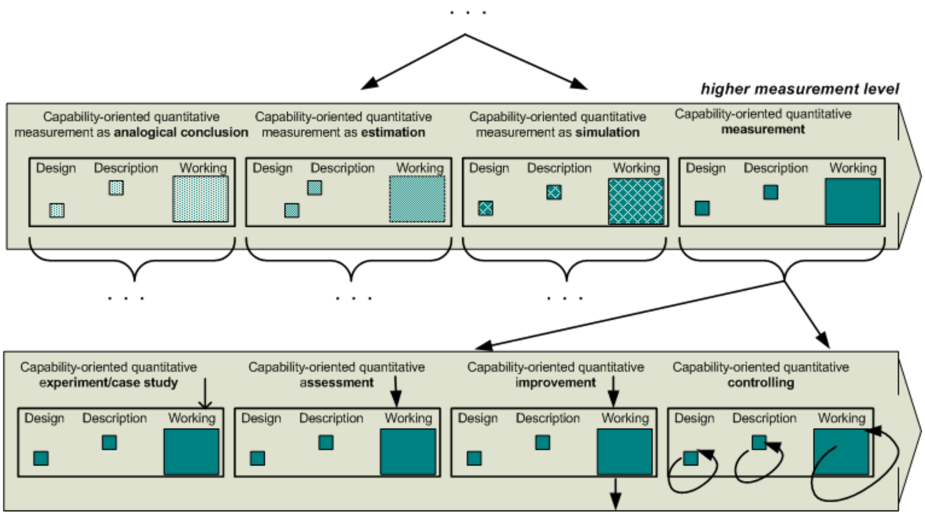

. . .
higher measurement level
Capability-oriented quantitative measurement as analogical conclusion
Capability-oriented quantitative measurement as estimation
Capability-oriented quantitative measurement as simulation
Capability-oriented quantitative measurement
Design
Description
Working
. . .
Capability-oriented quantitative experiment/case study
Capability-oriented quantitative assessment
Capability-oriented quantitative improvement
Capability-oriented quantitative controlling

Fig. 12: The CAME level of measurement process integration

Migration principles: (are based on (1.2), (2.3), (2.8), (2.11), (2.18), (2.19))

In order to achieve a full measurement process is a complex never ending task. Some helpful compromises could be based on the use of "lower level" measurement and evaluation approaches like estimation or analogies. This intention is characterized in following figure that shows the *Magdeburg Layer Model* (MLM) which means an appropriate use of measurement methods.

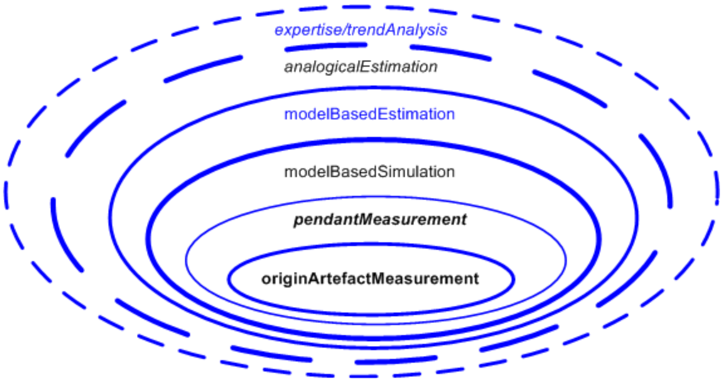


Fig. 13: The Magdeburg Layer Model of measurement alternatives

Furthermore, measurement migration involves the principles of integration of the measurement in the given IT area.

Efficiency characteristics:

Our intention of efficiency is mainly based on the kind of automation and defines the ***efficiency level.*** Therefore we differ between manual, semi automated (tool support) and automated ((monolithic) tools, open measurement components and measurement infrastructures). Otherwise the tool support could be based on an internal application or an external (outsourced respectively) tool realization.

Efficiency principles: (are based on (1.2), (2.8), (2.19))

The situation of *weak tool support* could be lessening by the use of semi-automatic or external CAME facilities. The following figure 14 considering the operational part of the CAME framework also.

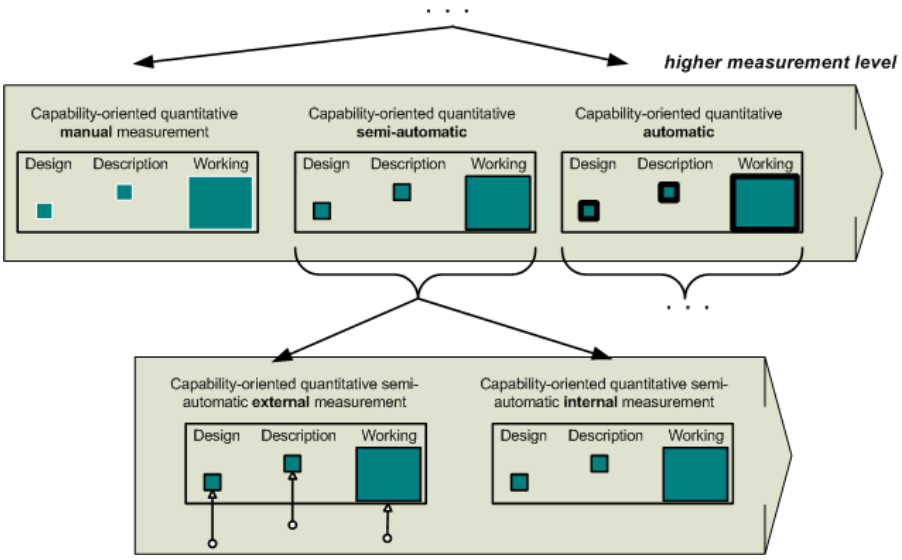

. . .
higher measurement level
Capability-oriented quantitative manual measurement
Design
Description
Working
Capability-oriented quantitative semi-automatic
Design
Description
Working
Capability-oriented quantitative automatic
Design
Description
Working
. . .
Capability-oriented quantitative semi-automatic external measurement
Design
Description
Working
Capability-oriented quantitative semi-automatic internal measurement
Design
Description
Working

Fig. 14: CAME framework based measurement characteristics

Finally, our CAME approach as a set of software measurement principles could be presented in a formulary manner based on the description above as following:

CAME STRATEGY:

$$CAME^{community}_{strategy} : \ \exists\ measurementCommunity \subset ITArea \qquad (3.1)$$

$$CAME^{accep\tan ce}_{strategy} : companyManagement(measurementAcceptance) \rightarrow \text{true} \qquad (3.2)$$

$$CAME^{motivation}_{strategy} : importance(firstMeasurementApplications) \rightarrow max \qquad (3.3)$$

$$CAME^{engagement}_{strategy} : realizedEffort(measurement) \geq measurementProcessContinuity \qquad (3.4)$$

CAME FRAMEWORK:

$$\boldsymbol{CAME}^{choice}_{framework}\ \mathbf{:}\ \boldsymbol{CAME}^{aspect-orientedMeasurement}_{choice}\ \mathbf{:}\quad measurement(product_{aspects} \vee process_{aspects} \vee resources_{aspects})$$

$$= \{CAME^{unders\tan ding}_{aspect-orientedMeasurement}, CAME^{evaluation}_{aspect-orientedMeasurement}, CAME^{managing}_{aspect-orientedMeasurement}\} \tag{3.5}$$

$$\boldsymbol{CAME}^{capability-orientedMeasurement}_{choice}\ \mathbf{:}\quad measurement(product \vee process \vee resources)$$

$$=\{CAME^{unders\tan ding}_{capability-orientedMeasurement}, CAME^{evaluation}_{capability-orientedMeasurement}, CAME^{managing}_{capability-orietedMeasurement}\} \tag{3.6}$$

$$\boldsymbol{CAME}^{totalMeasurement}_{choice}\ \mathbf{:}\quad measurement(product \wedge process \wedge resources)$$

$$=\{CAME^{unders\tan ding}_{totalMeasurement}, CAME^{evaluation}_{totalMeasurement}, CAME^{managing}_{totalMeasurement}\} \tag{3.7}$$

$$\boldsymbol{CAME}^{adjustment}_{framework}\ \mathbf{:}\boldsymbol{CAME}^{qualitativeMeasurement}_{adjustment}\ \mathbf{:}\quad metrics \in \{nominalScale \vee ordinalScale\}$$

$$=\{CAME^{tuned}_{qualitativeMeasurement}, CAME^{approximative}_{qualitativeMeasurement}\} \tag{3.8}$$

$$\boldsymbol{CAME}^{quantitativMeasurement}_{adjustment}\ \mathbf{:}\quad metrics \in \{intervalScale \vee atioScale\}$$

$$=\{CAME^{tuned}_{quantitativeMeasurement}, CAME^{approximative}_{quantitativeMeasurement}\} \tag{3.9}$$

$$\boldsymbol{CAME}^{migration}_{framework}\ \mathbf{:}\ \boldsymbol{CAME}^{ana\log icalConclusion}_{migration}\ \mathbf{:}\quad measurement(artefact') \rightarrow measurement(artefact)$$

$$=\{CAME^{\exp eriment/caseStudy}_{ana\log icalConclusion}, CAME^{assessment}_{ana\log icalConclusion}, CAME^{improvement}_{ana\log icalConclusion}, CAME^{controlling}_{ana\log icalConclusion}\} \tag{3.10}$$

$$\boldsymbol{CAME}^{estimation}_{migration}\ \mathbf{:}\quad estimationFunction(measurement(artefact_{aspects})) \rightarrow measurement(artefact)$$

$$=\{CAME^{\exp eriment/caseStudy}_{estimation}, CAME^{assessment}_{estimation}, CAME^{improvement}_{estimation}, CAME^{controlling}_{estimation}\} \tag{3.11}$$

$$\boldsymbol{CAME}^{simulation}_{migration}\ \mathbf{:}\ simulationModel(artefact) \rightarrow measurement(artefact)$$

$$=\{CAME_{simulation}^{\exp eriment/caseStudy}, CAME_{simulation}^{assessment}, CAME_{simulation}^{improvement},$$
$$CAME_{simulation}^{controlling}\} \tag{3.12}$$

$$\boldsymbol{CAME_{migration}^{measurement}} : measurement(artefact)$$
$$=\{CAME_{measurement}^{\exp eriment/caseStudy}, CAME_{measurement}^{assessment}, CAME_{measurement}^{improrvement},$$
$$CAME_{measurement}^{controlling}\} \tag{3.13}$$

$$\boldsymbol{CAME_{framework}^{efficiency}} : \boldsymbol{CAME_{efficiency}^{semi-automaticMeasurement}} :$$
$$CAMETool(measurementProcess_{phases})$$
$$=\{CAME_{semi-automaticMeasurement}^{\mathrm{int}\,ernal}, CAME_{semi-automaticMeasurement}^{external}\} \tag{3.14}$$

$$\boldsymbol{CAME_{efficiency}^{automaticMeasurement}} : CAMETool(measurementProcess)$$
$$=\{CAME_{automaticMeasurement}^{\mathrm{int}\,ernal}, CAME_{automaticMeasurement}^{external}\} \tag{3.15}$$

Note that a framework application involves all the chosen part of the bracket elements of the CAME framework description.

Furthermore, the CAME approach could be used in order to evaluate the measurement level itself in the following manner:

$$choiceLevel = |A_{measuredKindsOfProducts}| + |A_{measuredKindsOfProcesses}| + |A_{easuredKindsOfResources}| \tag{3.16}$$

We establish *choiceLevel = {1, 2, …, n}* where *n* depends on the metrication structure of the different measurement areas (product, process, resources). The following adjustment level considers the part of quantified measurement of the used measures for the different measurement areas characterized above.

$$adjustmentLevel = (|V^{(ratioScale)}| + |V^{(intervalScale)}|)/choiceLevel \tag{3.17}$$

The *migrationLevel* considers the measured areas and their implementation as a full measurement, evaluation and application process. Therefore we define

$$migrationLevel = |M^{(product)}| + |M^{(process)}| + |M^{(resources)} \tag{3.18}$$

The *toolLevel* could be simple evaluated by

$$toolLevel = \quad (|\, M^{(toolBased)} \,|)/migrationLevel \tag{3.19}$$

where A are the measured artefacts, M the different measurement and evaluation methods, and V the measurement values. The CAME approach is very helpful in order to analyze the general measurement situation in different IT areas, development paradigms and kinds of software systems and can establish necessary research activities and missing measurement field applications.

4 Web-Based Software Measurement Frameworks

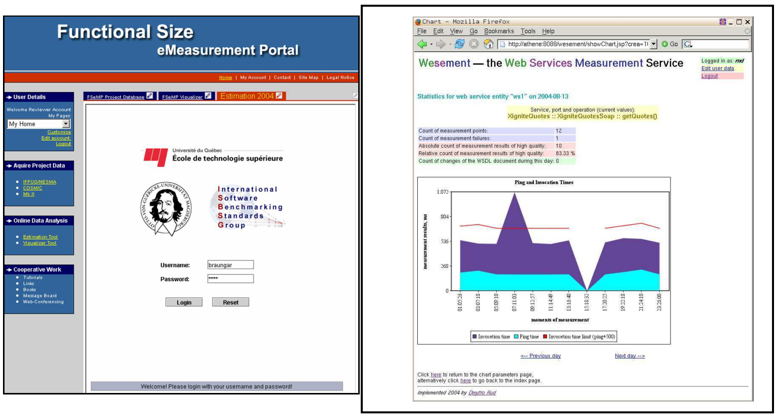
Based on the incremental Web use and the technologies of mobility, software quality assurance can be supported by ***e-measurement services***. This means that the software quality process would be divided in sub processes and subcomponents that are available in the Web. In order to support the implementation of measurement processes, *e-measurement consulting* (including an *e-measurement certification*) could be helpful [2].

Basic e-measurement principles are:

- The *ubiquitous characteristics* (including availability and mobility) of the resources as measurement artifacts and methods/services (4.1)
- The possibility of external supports by the use of different kinds of shoring (outsourcing, off shoring etc.) (4.2)
- The effects of the *pervasiveness* in the WWW considering the knowledge and application resources (4.3)
- The future principles of *self-managing* and *adaptation* of Web-based solutions and infrastructures (4.4)

Web-based measurement frameworks must be a ***living system*** like the World Wide Web itself. The realization of e-measurement in the World Wide Web leads to many kinds of infrastructures that combine the application, the consulting, the communication and the information ([5], [8]). In following e-measurement areas are described. The application of software measurement in the IT area does not have an appropriate implementation in many companies. Therefore, the creation and the hosting of *e-measurement communities* should be helpful in order to improve this current situation (see MAIN: http://www.mai-net.org/home/, ISERN: http://www.iese.fhg.de/ ISERN/, and DASMA: http://www.dasma.org/). These kinds of measurement support are based on the CAME principle (3.1) involving the Web characteristics (4.2) and (4.3).

Services or measurement realization and the presentation of the measurement results are meaningful supports and could be provided by special companies on the Web. Further, the collaboration on the Web between companies could produce some new kinds or levels of *e-measurement services*. Examples of this kind of services are shown in the figure 15 as a Functional Size Measurement Portal (FSeMP) [8] and a Web Service Trust Center (WSTC) [10] with a service of continuous performance measurement of real web services (see http://fsemp.cs.uni-magdeburg.de/ and http://ws-trust.cs.uni-magdeburg.de/). These kinds of measurement support are based on the CAME principles (3.3), (3.5), (3.8), (3.9), (3.13) and (3.14) involving the Web characteristics (4.1) and (4.4).



Welcome Reviewer Account

Username: braungar

Password: ****

Welcome! Please login with your username and password!

http://athene:8088/wesement/showChart.jsp?crea=1(

Wesement — the Web Services Measurement Service

Logged in as: md

Statistics for web service entity "ws1" on 2004-08-13

Service, port and operation (current values):
XigniteQuotes :: XigniteQuotesSoap :: getQuotes()

Count of measurement points:	12
Count of measurement failures:	1
Absolute count of measurement results of high quality:	10
Relative count of measurement results of high quality:	83.33 %
Count of changes of the WSDL document during this day:	0

Click here to return to the chart parameters page, alternatively click here to go back to the index page.

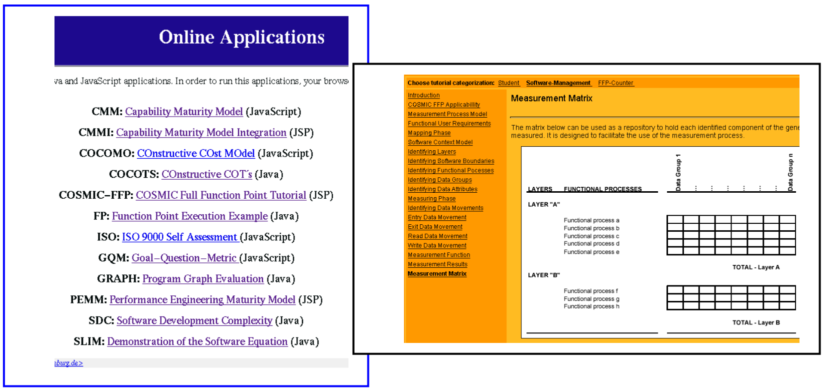
Implemented 2004 by Dmytro Rud

Fig. 15: Examples of e-measurement services

These kinds of services include consulting during the measurement planning, the measurement performance and the measurement exploration processes. Especially, the comparison of measurement results between companies or the discussion about measurement results could be supported by some kinds of *e-experience* and/or *e-repositories* in the World Wide Web. Fundamental knowledge about software measurement is an essential part of a successful software measurement process.

Hence, some kinds of *measurement e-learning* would be helpful managing these tasks. The e-learning services of the Software Measurement Laboratory of the University of Magdeburg (SML@b) [11] shown in figure 16 is one of the selected examples (on the left side the overview and on the right side the COSMIC Full Function Point tutorial). These kinds of measurement support are based on the CAME principles (3.4), (3.7), (3.13) and (3.14) involving the Web characteristics (4.1) and (4.2).

An OWL-based presentation of the ISO 15939 approach is founded in [4] and can be used for semantic Web-based measurement infrastructures which improves the measurement process level (as CAME migration) support and are based on the CAME principles (3.10) to (3.13) involving the Web characteristics (4.3) and (4.4).



Online Applications

va and JavaScript applications. In order to run this applications, your brows

CMM: Capability Maturity Model (JavaScript)

CMMI: Capability Maturity Model Integration (JSP)

COCOMO: COnstructive COst MOdel (JavaScript)

COCOTS: COnstructive COT´s (Java)

COSMIC–FFP: COSMIC Full Function Point Tutorial (JSP)

FP: Function Point Execution Example (Java)

ISO: ISO 9000 Self Assessment (JavaScript)

GQM: Goal–Question–Metric (JavaScript)

GRAPH: Program Graph Evaluation (Java)

PEMM: Performance Engineering Maturity Model (JSP)

SDC: Software Development Complexity (Java)

SLIM: Demonstration of the Software Equation (Java)

:burg.de>

Choose tutorial categorization: Student Software-Management FFP-Counter
Introduction
COSMIC FFP Applicabillity
Measurement Process Model
Functional User Requirements
Mapping Phase
Software Context Model
Identifying Layers
Identifying Software Boundaries
Identifying Functional Pocesses
Identifying Data Groups
Identifying Data Attributes
Measuring Phase
Identifying Data Movements
Entry Data Movement
Exit Data Movement
Read Data Movement
Write Data Movement
Measurement Function
Measurement Results
Measurement Matrix

Measurement Matrix

The matrix below can be used as a repository to hold each identified component of the gene measured. It is designed to facilitate the use of the measurement process.

Fig 16: Examples of measurement e-learning

5 Conclusions

This paper discussed the different characteristics and intentions of software measurement frameworks from a declarative and operational point of view. A measurement framework including strategic aspects is shown as a combination of the declarative and operational characteristics called *CAME strategy* and *CAME framework*. This software measurement framework improves the appropriateness of measurement by using alternatives, maturity successes and stepwise refinement.

Some helpful measurement direction of e-measurement was explained exemplary in order to combine the coverage of the whole software measurement processes with the current and future features of the Web-based technologies and infrastructures.

References

1. Abran, A., Bundschuh, M., Büren, G., Dumke, R. (eds.): Software Measurement-Research and Application. Proc. of the IWSM/Metrikon 2004, November 2004, Berlin, Shaker Publ., Aachen (2004)
2. Dumke, R. R.: Software Measurement Frameworks. Proc. of the 3rd World Congress for Software Quality, Munich, Sept. 2005, Vol. III, pp. 75-84
3. Dumke, R.; Foltin, E.: An Object-Oriented Software Measurement and Evaluation Framework. Proc. of the FESMA, October 4-8, Amsterdam (1999) pp. 59-68
4. Dumke, . R.; Kunz, M.; Hegewald, H.; Yazbek, H.: An Agent-Based Measurement Infrastructure. Proc. of the IWSM 2005, Shaker Publ., pp. 415-434
5. Dumke, R.; Schmietendorf, A.; Zuse, H.: Formal Description of Software Measurement and Evaluation – A Short Overview and Evaluation. Preprint No 4,

University of Magdeburg, Faculty of Informatics, (see http://ivs.cs.uni-magdeburg.de/sw-eng/agruppe/forschung/paper/FormalM.pdf) (2005)

6. Ebert, C.; Dumke, R.; Bundschuh, M.; Schmietendorf, A.: Best Practices in Software Measurement. Springer Publ. (2004)
7. ISO/IEC 15939: Information Technology – Software Measurement Process. Metrics News 6(2001) pp. 11-46
8. Lother, M.; Braungarten, R.; Kunz, M.; Dumke, R.: The Functional Size eMeasurement Portal (FSeMP). In: [1] pp. 27-40
9. Pandian, C. R.: Software Metrics – A Guide to Planning, Analysis, and Application. CRC Press (2004)
10. Schmietendorf, A.; Dumke, R.; Reitz, D.: SLA Management – Challenges in the Web-Service-Based Infrastructures. Proc. of the ICWS 2004, San Diego (2004) 606-613
11. SML@b Web Site: On-line in http://ivs.cs.uni-magdeburg.de/sw-eng/us/
12. Solingen, v. R.; Berghout, E.: The Goal/Question/Metric Method. McGraw Hill Publ. (1999)
13. Tayntor, C. B.: Six Sigma Software Development. CRC Press (2003)
14. Wille, C.: Software Agent Measurement Framework. Shaker Publ., Aachen (2005)
15. Zuse, H.: A Framework of Software Measurement. de Gruyter Publ., Berlin (1998)

Estimating the performance and capacity of software processes according to ISO/IEC 15504

Francisco J. Pino[1,2], Félix García[2], Manuel Serrano[2], and Mario Piattini[2]

[1] IDIS Research Group
Electronic and Telecommunications Engineering Faculty
University of the Cauca
Street 5 # 4 – 70, Popayán, Colombia.
fjpino@unicauca.edu.co
[2] ALARCOS Research Group
Information Systems and Technologies Department
UCLM-Soluziona Research and Development Institute
University of Castilla-La Mancha
Paseo de la Universidad, 4 – 13071 Ciudad Real, Spain
{Felix.Garcia, Manuel.Serrano, Mario.Piattini}@uclm.es

Abstract. At the moment there is no set of metrics which measures the improvements brought in by efforts to make software processes better. It is often the case that these improvements are measured using informal and subjective processes based on the perception of employees and/or auditors. Bearing all this in mind, this work presents a set of measurements for gauging the performance and capability of software processes, based on the international standard ISO/IEC 15504. This set of metrics aims to lower the level of subjectivity of people when measuring the processes. A more objective and hence more formal evaluation is thus achieved.

1 Introduction

Nowadays, software companies know that success (in terms of time, money, quality, etc.) in delivering a product lies in an effective management of its software processes [6], which involves four key responsibilities [9]: (i) process definition, (ii) process measurement, (iii) process control and (iv) process improvement.

One of the main reasons for the massive increase in the interest in software measurement is the perception that measuring the quality of the improvement process [7] is another crucial activity. This involves carrying out an efficient and effective measurement process, with the following main goals: (i) to help in understanding the development and to maintain tasks, (ii) to allow projects to be controlled, (iii) to enhance our processes and products [8].

However, when dealing with process measurement, it is commonly known that, generally speaking, most measures are defined for products: measures for software processes are scarce. It is therefore important to devote our efforts towards research into software process measurement. This is a key activity for the success of software process management and improvement, as this kind of activity, which gives some feedback to the process, depends on an appropriate and objective measurement of that process.

The importance of process management justifies many of the standardization initiatives of process improvement, such as CMM, Bootstrap or SPICE. Also, by measuring the capacity of processes we can estimate the maturity of the organization, as stated in current international standards such as CMMI [2] or ISO 15504 [4] [5] which are widely accepted and used.

Currently, process improvement is measured by informal and subjective processes based upon the perception of employees and evaluators. Unfortunately, they are not based on formal measurement processes [11]. In this work, we present a set of measures which are designed to evaluate the performance and capacity of software processes, following the international standard ISO/IEC 15504. With this set of measures we aim to lower the level of subjectivity when measuring processes, increasing the formality and objectivity of the evaluation.

2 Framework for metrics definition

International standards related to evaluation methods present a general framework for evaluating and defining several indicators that must be taken into account when performing an evaluation. However, they do not define explicit measures that help us in calculating the performance or capacity of a process. This value is very important when we are trying to evaluate the maturity of the company, as this is closely related to the capacity of their processes.

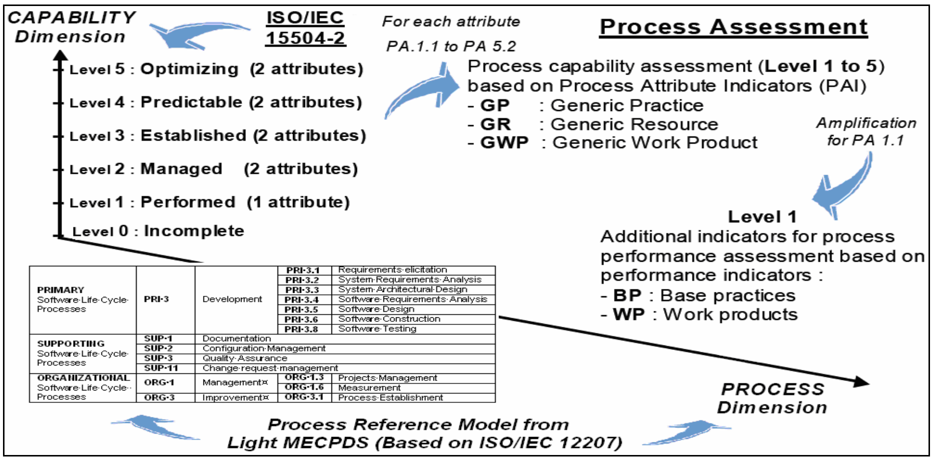
The scope of the current work is summarised in the following:

- Regarding the method for measure construction, we will apply the method proposed in [13].
- As framework for the software process evaluation model, our measurement proposal will be based on the ISO/IEC 15504 standard (see figure 1). More specifically, we will focus on levels 1 (performed) and 2 (managed).
- We will use the set of processes defined by Light MECPDS [12] as a reference model, which is based on the international standard ISO/IEC 12207:2004 [3], (see figure 1).
- The measures defined can be directly used by the Light MECPDS evaluation method, but can be adopted by any other model based on ISO/IEC 15504.

As we can observe in figure 1, two kinds of measures are proposed:

- The first kind is related to the capacity dimension, and its goal is to measure the capacity of a process, taking into account the process attributes related to the capacity levels defined by ISO/IEC 15504. For each process attribute, we will define a "capacity measure" based on the measurement of the following indicators: (i) generic practice, (ii) generic resources used and (iii) generic work products obtained in the process. These indicators are based on the ISO/IEC 15504-5Standard.
- The second group of measures is related to the process dimension, and their goal is to measure the process performance by considering the characteristics of the

processes defined in the Process Reference Model of Light MECPDS. For each sub-process, the “performance measure” is based on the following indicators (which have been obtained from the ISO/IEC 15504-5 e ISO/IEC 12207 standards): (i) performed base practices and (ii) obtained work products.


CAPABILITY
Dimension
ISO/IEC
15504-2
For each attribute
PA.1.1 to PA 5.2
Process Assessment
Level 5 : Optimizing (2 attributes)
Level 4 : Predictable (2 attributes)
Level 3 : Established (2 attributes)
Level 2 : Managed (2 attributes)
Level 1 : Performed (1 attribute)
Level 0 : Incomplete
Process capability assessment (Level 1 to 5)
based on Process Attribute Indicators (PAI)
- GP : Generic Practice
- GR : Generic Resource
- GWP : Generic Work Product
Amplification
for PA 1.1
Level 1
Additional indicators for process
performance assessment based on
performance indicators :
- BP : Base practices
- WP : Work products
PROCESS
Dimension
Process Reference Model from
Light MECPDS (Based on ISO/IEC 12207)

PRIMARY Software·Life·Cycle·Processes	**PRI·3**	Development	**PRI·3.1**	Requirements·elicitation
			PRI·3.2	System·Requirements·Analysis
			PRI·3.3	System·Architectural·Design
			PRI·3.4	Software·Requirements·Analysis
			PRI·3.5	Software·Design
			PRI·3.6	Software·Construction
			PRI·3.8	Software·Testing
SUPPORTING Software·Life·Cycle·Processes	**SUP·1**	Documentation		
	SUP·2	Configuration·Management		
	SUP·3	Quality·Assurance		
	SUP·11	Change·request·management		
ORGANIZATIONAL Software·Life·Cycle··Processes	**ORG·1**	Management¤	**ORG·1.3**	Projects·Management
			ORG·1.6	Measurement
	ORG·3	Improvement¤	**ORG·3.1**	Process·Establishment

Fig. 1. Structure and indicators for measure definition.

2.1 Need for Information

When carrying out a process evaluation in a company context, we need to follow an evaluation method that generates quantitative results which characterize the performance and capacity of the process (or the organization’s maturity) [3, 10]. These results give information that allow us to determine the current state of the software process so we can find the strengths and weaknesses that allow us to define strategies for enhancing the processes.

To obtain the relevant information about a process performance and capacity, it is necessary to provide a set of measures that allow the evaluation processes to work in a way that is both more formal and more objective.

2.2 Goal of Metrics

We have used the GQM method to define clearly the goal that we want to reach by using the proposed metrics. The next table shows the general goal we want to achieve.

Table 1. Goal definition

GOAL	
To analyze	The software process
With the purpose of	Evaluating
With respect to	The performance and capacity
From the point of view of	The improvement process group
In the context of	International Standard ISO/IEC 15504-5:2006(E)

3 Definition of performance process metrics

We have analyzed a standard process from the ISO/IEC 15504-5:2006 standard to define the metrics at level 1 or performance level. As all the processes that follow the standard have the same structure, we can define the metrics for the other processes of the reference model based on the one presented here. Figure 2 shows the structure of the quality assurance process, which we have used as a base for the definition of the metrics for measuring process performance.

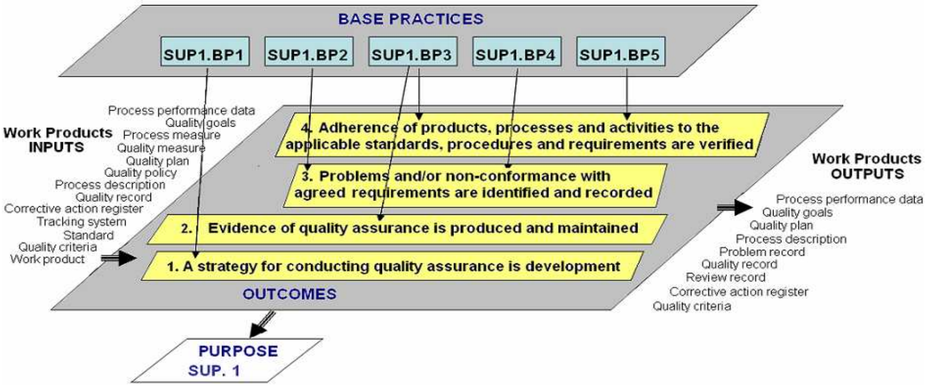


Fig. 2. Structure of the Quality Assurance process in ISO/IEC 15504-5:2006

3.1 Questions

As a starting point in the performance metrics definition process, we have stated a set of hypotheses about the software processes. We have defined these hypotheses as questions we want to answer on the path to getting a valid set of processes metrics.

- Does the achievement of the base practices influence the results of the software process?
- Do the input work products influence the results of the software process?
- Do the output work products influence the results of the software process?
- Do the results of a process influence the performance of a software process?

3.2 Measures Definition

As set out in the schema of the Standard ISO/IEC 15504-5:2006, the process performance can be measured by means of the successful implementation of the results. These results are related to the base practices and work products.

The measures of the level of process performance have been defined, with the aim of evaluating the degree of process fulfillment with respect to the process defined in the process evaluation model. The measure definition is shown in Table 2:

Table 2. Process Performance Measures

Process Performance Measures	
1. Based on Base Practices	
Measure	**Definition**
NRP_std	Number of results (defined in ISO/IEC 15504-5) of the software process being evaluated.
NBPRi_std	Number of base practices (defined in ISO/IEC 15504-5) which contribute to the achievement of the result *i* of the software process being evaluated.
WRP	Weight of each result of the software process being evaluated **WRP = 1 / NRP_std**
VBPRi_ro	Value of the base practices for the result *i* achievement carried out by the organization. ***It is obtained from an information collection tool***.
DFRi (BP)	Degree of fulfillment of the result *i* according to the base practices. **DFRi (BP) = VBPRi_ro / NBPRi_std**
DPP (BP)	Degree of Process Performance based on the base practices. $\textbf{DPP (BP) = WRP} * \sum_{i=1}^{n} \textbf{DFR}\boldsymbol{i}\textbf{ (BP)}$
2. Based on Work Products	
Measure	**Definition**
NIWPRi_std	Number of input work products of the software process being evaluated (defined in ISO/IEC 15504-5) related to the result *i*.
NOWPRi_std	Number of output work products of the software process being evaluated(defined in ISO/IEC 15504-5) related to the result *i*.
TNWP_Ri	Total number of work products of the result *i*. **TNWP_Ri = NIWPRi_std + NOWPRi_std**
NWPRi_ro	Number of work products carried out by the organization for the result i achievement. **It is obtained from an information collection tool.**
DFRi (WP)	Degree of fulfillment of the result *i* according to the work products. **DFRi (WP) = NWPRi_ro / TNWP_Ri**
DPP (WP)	Degree of Process Performance based on work products. $\textbf{DPP (WP) = WRP} * \sum_{i=1}^{n} \textbf{DFR}\boldsymbol{i}\textbf{ (WP)}$

The process results defined in ISO/IEC 15504-5 are related on the one hand to base practices and on the other to work products. So, in order to obtain a solid measure of process performance, there is an undergirding premise that both the base practices and work products have the same weight.

Table 3. Process Performance Measure

Process Performance Measures	
Based on Base Practices and Work Products	
Measure	**Definition**
GPPM	Global Process Performance Measure **GPPM = DPP (BP) * 0.5 + DPP (WP) * 0.5**

4 Process Capability Measure Definition

We have analyzed a capability level from the ISO/IEC 15504-5:2006 standard for defining the metrics at level 2 or capability level. As all the capability levels of the Standard have the same structure, based on the defined measures for level 2, the measures of upper levels can be obtained. The capability level chosen in the context of this paper has been the level 2, “Managed Process” [4], whose structure is illustrated in Figure 3.

It is important to highlight that every process attribute result has only one generic practice associated, as well as generic resources and generic work products which are related to these results.

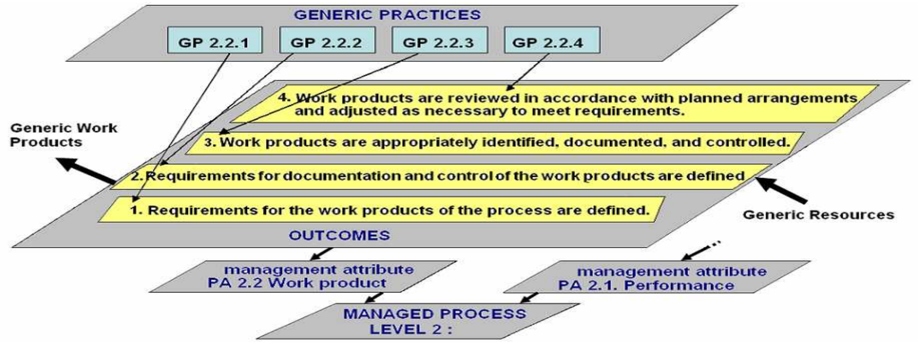

GENERIC PRACTICES
GP 2.2.1
GP 2.2.2
GP 2.2.3
GP 2.2.4
4. Work products are reviewed in accordance with planned arrangements and adjusted as necessary to meet requirements.
Generic Work Products
3. Work products are appropriately identified, documented, and controlled.
2. Requirements for documentation and control of the work products are defined
1. Requirements for the work products of the process are defined.
Generic Resources
OUTCOMES
management attribute PA 2.2 Work product
management attribute PA 2.1. Performance
MANAGED PROCESS LEVEL 2 :

Fig. 3. Capability Level 2 Structure of the ISO/IEC 15504-5:2006

4.1 Questions

As a starting point for the capability metrics definition process, we have stated a set of hypotheses about the software processes. We have defined these hypotheses as questions we want to answer as we go towards getting a valid set of process metrics.

- Do the process attributes affect the obtaining of a capability level?
- Do the results of a process influence the software process capability?
- Do the generic resources affect the software process capability?
- Do the generic work products affect the software process capability?
- Does the achievement of the generic practices have any influence on the results of a software process attribute?

4.2 Measures Definition

According to the schema of the Standard ISO/IEC 15504-5:2006, the process capability can be measured through the successful implementation of the process attributes. These process attributes are related to the generic practices, resources and work products. The measures at the level of the scope of the process capability have been defined with the aim of evaluating the capability level of the process with respect to a capability model. The definition of these measures is provided in Table 4.

Table 4. Measures of the Process Capability Attribute

Measures of the Process Capability Attribute	
1. Based on Generic Practices	
Measure	**Definition**
NARP_std	Number of attribute results (defined in ISO/IEC 15504-5) of the process being evaluated.
NGPRi_std	Number of generic practices (defined in ISO/IEC 15504-5) of the process attribute being evaluated which contribute to the achievement of the result *i*
WRAP	Weight of each result of the attributes of the software process being evaluated **WRAP = 1 / NARP_std**
VGPRi_ro	Value of the generic practices carried out by the organization for the result *i* achievement. ***It is obtained from an information collection tool***.
DFRi (GP)	Degree of fulfillment of the result *i*, according to the generic practices. **DFRi (GP) = VGPRi_ro / NGPRi_std**
DPAF (GP)	Degree of Process Attribute Fulfillment based on generic practices **DPAF (GP) = WRAP *** $\sum_{i=1}^{n}$ **DFR*i* (GP)**
2. Based on Generic Resources	
Measure	**Definition**
NGRRi_std	Number of generic resources of the software process attribute being evaluated (defined in ISO/IEC 15504-5) related to the result *i*.
NGRRi_ro	Number of generic resources which are available in the organization for the result *i*. ***It is obtained from an information collection tool***.
DFRi (GR)	Degree of fulfillment of the result *i* according to the generic resources. **DFRi (GR) = NGRRi_ro / NGRRi_std**
DPAF (GR)	Degree of Process Attribute Fulfillment based on generic resources **DPAF (GR) = WRAP *** $\sum_{i=1}^{n}$ **DFR*i* (GR)**
3. Based on Generic Work Products	
Measure	**Definition**
NGWPRi_std	Number of generic work products (defined in ISO/IEC 15504-5) of the process attribute to evaluate which contribute to the achievement of the result *i*
NGWPRi_ro	Number of the generic work products for the result *i*, actually carried out by the organization ***It is obtained from an information collection tool***.
DFRi (GWP)	Degree of fulfillment of the result *i* according to the generic work products **DFRi (GWP) = NGWPRi_ro / NGWPRi_std**
DPAF (GWP)	Degree Process Attribute Fulfillment based on generic work products. **DPAF (GWP) = WRAP *** $\sum_{i=1}^{n}$ **DFR*i* (GWP)**

The process attribute results defined in ISO/IEC 15504-5 are related to the generic practices, resources and work products. So in order to obtain a solid measure for the process capability, the weight for all these indicators is considered (see Table 5).

Table 5. Process capability measures

Capability Process Measures	
Based on Process Attributes	
Measure	**Definition**
GCPM	Global Capability Process Measure. **GCPM = DPAF (GP) * 0.4 + DPAF (GR) * 0.3 + DPAF (GWP) * 0.3**

5 Measures support tool

Once the measures were defined, a tool based on Bayesian Networks, supported in the Elvira Software Tool [1], for information collection and automatic calculation was developed. The measures had to be collected for all the process attributes in the capacity dimension and for each process in the process dimension. The aim of this tool is to provide companies with a useful instrument to automate the measurement process and to reduce the subjectivity of the evaluation process. The prototype window that supports the calculation of the performance metrics of the quality assurance process is shown in fig. 4.

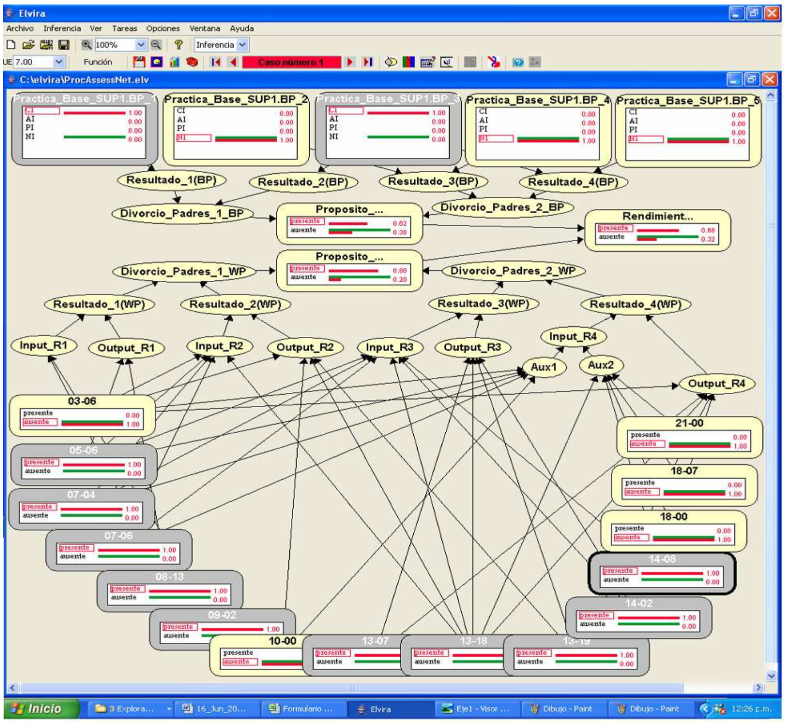


Fig. 4. Bayesian Networks for automatic calculation of the metrics.

6 Conclusions and future work

This work is complementary to the standard. The standard offers a horizontal view of the measurement process, because it provides the main threads in the evaluation of software processes. The work we have presented in this paper is vertical to the measurement process, however, as it provides metrics and forms for information gathering that help us to evaluate a software process in a formal and objective way.

A software company in search of process maturity should be disciplined in software measurement. If we use a process oriented focus, the need is not only to measure the product. It is also necessary to be able to measure the processes for improving the quality of the software product. The goal is to improve the quality of the software product built by the company, by raising the efficiency and effectiveness of the organizational process. Its competitivity in the global market will be heightened correspondingly. To enhance the processes it is crucial to measure appropriately, so in this work we set out to provide companies with easier and more objective processes for measuring and evaluating their processes, and to make the measurement more objective.

We have to take into account that the international standards, as far as evaluation methods are concerned, define a general framework for carrying out the evaluation, but they do not define explicit measures that help in determining the performance value or the process capacity. This value is very important when trying to assess the company's maturity, as organizational maturity is closely linked to the capacity of company processes.

In this work we have also developed some forms for information gathering. These forms are simple and by using them we can obtain valuable information for assessing the performance and capacity of the process being evaluated. We have also built a Bayesian net-based tool to facilitate the collection of information and the calculation of the value of metrics.

Taking this work as a starting point, we have seen several future lines of work:

- To define the weights of the performance metric coefficients for base practices and work products, using studies that have been carried out in this field as a basis.
- To define the weights of the capacity metric coefficients for generic practices, generic resources and generic work products, based on studies that have been carried out in this field.
- To analyze the relationship between base practices and work products in the ISO/IEC 15504:2006 standard.

Currently, these measurements are being used in two software enhancement programs in two small companies from the south-western part of Colombia (named SIDEN Ltda. y Unisoft Colombia Ltda.) the purpose being to validate and refine those metrics.

Acknowledgments

This research is part of the following projects: “Mejora Basada en Evidencia de la Capacidad en Actividades de Software – MECENAS” (PBI06-0024) granted by “Junta de Comunidades de Castilla-La Mancha” and “Process Improvement for Promoting Iberoamerican Software Small and Medium Enterprises Competitiveness – COMPETISOFT” (506PI287) financed by CYTED.

References

1. *Proyecto Elvira*. [cited July, 2006]; Available from: http://www.ia.uned.es/investig/proyectos/elvira/.
2. *CMMI for Systems Engineering/Software Engineering, Version 1.1*. 2002, Software Engineering Institute: Pittsburgh.
3. *ISO/IEC 12207:2002/FDAM 2. Information technology - Software life cycle processes*. 2004, International Organization for Standardization: Geneva.
4. *ISO/IEC 15504-2:2003/Cor.1:2004(E). Information technology - Process assessment - Part 2: Performing an assessment*. 2004, International Organization for Standardization: Geneva.
5. *ISO/IEC 15504-5:2006(E). Information technology - Process assessment - Part 5: An exemplar Process Assessment Model*. 2006, International Organization for Standardization: Geneva.
6. Derniame, J.-C., A.B. Kaba, and B. Warboys, *The Software Process: Modelling and Technology*, in *Software process: principles, methodology, and Technology*, C. Montenegro, Editor. 1999, Springer: Germany. p. 1-12.
7. Fenton, N., *Metrics for Software Process Improvement.*, in *Software Process Improvement: Metrics, Measurement and Process Modelling*, M. Haug, E.W. Olsen, and L. Bergman, Editors. 2001, Springer. p. 34-55.
8. Fenton, N. and S. Pfleeger, *Software Metrics: A Rigorous Approach*. 2nd ed. 1997, London: Chapman & Hall.
9. Florac, W.A., R.E. Park, and A.D. Carleton, *Practical Software Measurement: Measuring for Process Management and Improvement*. 1997, Pittsburgh: Software Engineering Institute, Carnegie Mellon University. 1-12.
10. IEEE, C.S., *Guide to the Software Engineering Body of Knowledge - SWEBOK*. 2004, Washington: Angela Burgess. 119-146.
11. Pino, F., F. García, and M. Piattini, *Revisión sistemática de mejora de procesos software en micro, pequeñas y medianas empresas.* Revista Española de Innovación Calidad e Ingenieria del Software (REICIS), 2006. **2**(1): p. in press.
12. Pino, F., et al. *A Lightweight Model for the Assessment of Software Processes*. in *European Systems & Software Process Improvement and Innovation (EuroSPI 2006)*. Octuber, 2006. Joensuu, Finland: p. in press.
13. Serrano, M., C. Calero, and M. Piattini, *Metrics for Data Warehouse Quality*, in *Encyclopedia of Information Science and Technology (IV)*. 2005. p. 1938-1944.

General Recommendations on Increasing Efficiency of Inspections

Eugenia Egorova[1]

[1] Politecnico di Torino, Corso Duca degli Abruzzi 24,
10129 Torino, Italy
eugenia.egorova@polito.it

Abstract. Inspections have become integral part of the software development. The inspection process has many attributes that are easy to measure. Companies use these measurements as part of controlling and planning instruments. To improve review techniques, researchers are experimenting with attributes. This paper describes an analysis of industrial historical data. The nonparametric Mann-Whitney statistical method has been used to test number of hypotheses. Firstly it has been demonstrated that slower reading during preparation leads to higher Defect Rate. Secondly the experiment with number of reviewers in the group has been replicated. Previous studies have shown effectiveness of small teams in controlled conditions but we have proved its validity in the industry as a whole. Finally we have confirmed the value of industrial metrics and prepared basis for the controlled experiment in the industry.

Keywords: inspections, software metrics, preparation rate, defect rate, process improvement.

1 Introduction

Software producing companies invest great effort into delivery of high-quality products. Demands of the market require optimization of the production process. Many companies are trying to use best practices for the process improvement.

Software inspections have become one of the most adopted such practices. They allow early defect elimination from design and code, improving overall product and process quality. Starting from 1976 [1] and thoroughly discussed in literature, inspection process itself became subject of improvement.

Large part of the research has been done in laboratory conditions. Controlled experiments allow study of different factors that influence results of an inspection. Measurements have helped to analyze these results. Smaller part of the research has been made in industry. Results of these studies showed great importance of inspections in the overall software producing process. Additionally they have illuminated problems with actively used resources for increasing quality.

Current study is made on the industrial inspection data. It describes preparatory work for further measurement-based industrial experiments oriented towards improvements of resource management. Two resources are in the light of attention:

people and time. Number of hypotheses was tested and results have proved importance of the total preparation time of the inspection team and limiting number of people in this team. These aspects have been studied before [4-7]. Contribution of the current work comes from deriving results with high statistical level of significance from industrial data.

The paper organized as follows: Section 2 summarizes related work. Section 3 gives overview of the data and metrics that are the basis for this research. Section 4 describes experiment and sets of hypotheses under study. Section 5 reports the results of the experiment. Section 6 discusses pros and cons of the experiment. Section 7 provides summary of the paper and outlines further work.

2 Related Work

Inspections have been systematically addressed in Fagan's report [1] in 1976. Another source to learn classical inspections process is the book by Gilb and Graham [2]. Both works describe in details every inspection's subprocess. Recent works study mainly either modifications of the original process or analyze various weak sides of the formal inspections.

Biffl and Gurjahr [5] found that optimal results can be achieved while combining number of inspections, roles and different reading techniques. They also underlined that after including an additional inspector in big team efficiency increasing has become insignificant. Halling and Biffl [6] further investigated influence of reading techniques and team size on the results of inspections. Results of the team size analysis have showed that there is no observable influence of the team size on the efficiency of inspections. The study emphasized importance of further research on correlation between team size and meeting results.

Another detailed investigation of combination of factors was done by Porter et al. [7]. Among other questions influence of the team size on duration and efficiency of an inspection was studied. The results have shown that “inspection interval and effectiveness of defect detection were not significantly affected by team size”.

Thus significant amount of studies has been made in order to analyze influence of team size on the efficiency of inspections. Most of them were made in the form of controlled experiments. We have taken again this topic to test situation in industry and see first if the number of inspections performed by large teams is significant and secondly to test industrial data as it is. The preparation effort was analyzed mainly as various reading techniques [5, 6]. In this work we have taken it as a time factor.

3 Data and Metrics

The data that became the basis for this research has been provided by a large software developing company. This data comes from the extensive data-base which is used by the company to control and plan their development process in order to insure its effectiveness and high quality of the products. Inspections are one the obligatory steps of the development process. This empirical study is done on the data containing code and design inspections during one year (2005). Every inspection was measured in terms of size of the artifact, preparation time, meeting time, number of major and minor errors, number of inspectors, and some other attributes. The company is satisfied with inspections performed as a part of their development process. Their intention is to find areas of the process improvement.

4 Empirical Study

During initial detailed study of the inspection data number of observations was noticed and examined. We have made assumptions about process parameters and tested them in the form of hypotheses. Without having background information about experience of the inspectors or quality of the documents we supposed that they are average across the organization. Second premise was that during preparation before the meeting all reviewers use same reading technique adopted in the company. Basis for these suppositions is discussed in the Section 6.

4.1 Data Preparation

Inspections' database contains information on different types of inspections: with no physical meeting when information is exchanged via emails, very short meetings, single session and multi-session meetings. For the current study we have selected inspections with meeting duration from 30 minutes to 6 hours. This limitation gave more evenly distributed data set.

Second constraint was for the results of the inspections. We have added value for total number of defects found during each inspection. For this study we have taken inspections with non-zero total number of defects.

Both code and design inspections are studied in this work. After applying constraints mentioned above we have 121 design inspections and 132 code ones.

Table 1. Summary for the inspections that have participated in the study.

Artifact type	*Minimum number of defects*	*Maximum number of defects*	*Minimum number of reviewers*	*Maximum number of reviewers*	*Minimum size under review*	*Maximum size under review*
Design	1	31	2	9	1 (pages)	92 (pages)
Code	1	20	2	12	1 (KLoc)	800 (KLoc)

4.2 Values Under Study

We have used weighted values to normalize the data across number of defects, number of reviewers, total preparation time and document size. These values are

Defect Rate and Preparation Rate. They are calculated for all inspections of the study according to the formulas 1 and 2.

$$\text{Defect Rate} = \text{Total Number of Found Faults} / \text{Document Size} . \tag{1}$$

$$\text{Preparation Rate} = \text{Document Size} / (\text{Total Preparation Time} / \text{Number of Reviewers}) . \tag{2}$$

4.3 Statistical Hypotheses and Statistical Method

In the study we have tested two sets of hypotheses. Each set consists of zero and alternative hypotheses:

- zero hypothesis (H0) proves absence of the statistical difference in a certain feature for two data sets;
- alternative hypothesis (H1) proves significance of the statistical difference in this feature.

All hypotheses' tests in this study are performed using nonparametric Mann-Whitney statistical method [10], with p=0,05. For this method H0 type of the hypotheses is accepted when empirical value (Uemp) is larger then the table value (Ucrit), otherwise H0 is rejected and H1 is accepted. The smaller is Uemp then Ucrit the more statistically different are two data sets in the tested feature.

4.4 Preparation Rate vs. Defect Rate

Assumption: higher Preparation Rate leads to lower Defect Rate.

Explanation: Preparation Rate is conventional name for average speed of checking [2, 3] an artifact by the team members before the meeting. Higher rate describes faster going through a document while lower rate represents more thoroughly performed preparation for an inspection.

Set 1 of the hypotheses under study:

H0: Level of the Defect Rate in average is the same for the inspections with different Preparation Rate.

H1: Level of the Defect Rate in average is lower for the inspections with higher Preparation Rate as compared to the inspections with lower Preparation Rate.

Variables:

- Independent value is Preparation Rate. To observe non-linear dependency between independent and dependent variables more then 2 gradations of the factor are needed. We have introduced 3 levels of the Preparation Rate:
  1) low (4-49 KLoc/hour or 2-19 pages/hour)
  2) medium (50-149 KLoc/hour or 20-39 pages/hour)

3) high (150-1000 KLoc/hour or 40-200 pages/hour)

- Dependent value is Defect Rate

Results of the hypotheses' testing are given in the Section 5.1.

4.5 Size of the Inspection Group vs. Defect Rate

Assumption: small group of inspectors (2-4 inspectors) in average finds the same number of errors as large group (5 or more inspectors).

Set 2 of the hypotheses under study:

H0: the level of the Defect Rate in average is the same for small and large groups.

H1: the level of the Defect Rate in average is higher for the large groups as compared to the small groups.

Variables:

- Independent value is size of the group
- Dependent value is Defect Rate

Results of the hypotheses' testing are given in the Section 5.2.

5 Results

We have studied 121 design and 132 code inspections. Two sets of hypotheses were tested with Mann-Whitney nonparametric statistical method. First we have tested if preparation time influences results of an inspection. We hypothesized that teams with longer preparation time in average have higher Defect Rate. Secondly we analyzed that in the industry team size can be limited without loss in quality.

5.1 Preparation Rate vs. Defect Rate

Statistical method Mann-Whitney is used to compare two samples. Since we have 3 levels for the Preparation Rate (low, medium and high), we have 3 data sets to compare. We have tested first set of the hypotheses (Section 4.4) comparing data sets pairwise: low with medium and medium with high.

Table 2 provides results for the design inspections. For both Test 1 (low vs. medium) and Test 2 (medium vs. high) Uemp is smaller then Ucrit. Results for the code inspections are given in Table 3. For both Test 3 (low vs. medium) and Test 4 (medium vs. high) Uemp is also smaller then Ucrit, the difference is even larger. For both types of inspections we reject H0 and accept H1. This supports the assumption that *design and code inspections with higher Preparation Rates in average result in lower Defect Rates.*

To see the general decreasing tendency we have plotted averages of all data sets (Figure 1 for design and Figure 2 for code inspections). For the design reviews the decreasing level is close to linear. For the code reviews the decreasing level has non-linear character but it is difficult to determine it on the small number data sets.

Table 2. Results of the hypotheses testing for the design inspections.

Test number	*Group 1*	*Group 2*	*Uemp*	*Ucrit, p = 0.05*
Test 1	PR = low	PR = medium	538,5	595
Test 2	PR = medium	PR = high	515	595

Table 3. Results of the hypotheses testing for the code inspections.

Test number	*Group 1*	*Group 2*	*Uemp*	*Ucrit, p = 0.05*
Test 3	PR = low	PR = medium	376	770
Test 4	PR = medium	PR = high	416	770

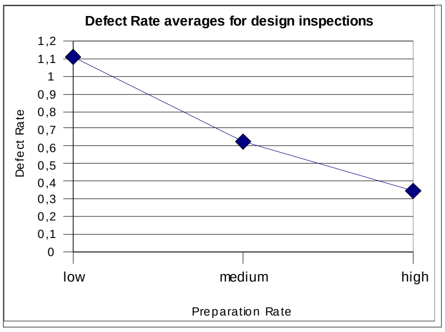


Fig. 1. Design inspections. Average Defect Rate is decreasing with growth of Preparation Rate from low to medium and from medium to high. Diminution has close to linear shape.

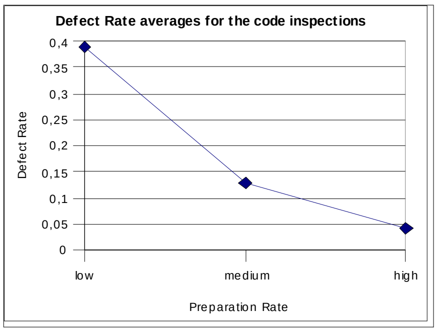


Fig. 2. Code inspections. Average Defect Rate is decreasing more steeply with growth of Preparation Rate from low to medium and from medium to high.

5.2 Size of the Inspection Group vs. Defect Rate

Second set of the hypotheses described in Section 4.5 is also tested with Mann-Whitney statistical method. Results are given in Table 4.

Table 4. Results of the hypotheses testing for the design and code inspections.

Artifact type	*Uemp*	*Ucrit, p = 0.05*
Design	492	471
Code	746,5	688

For both code and design data samples Uemp is larger then Ucrit. According to the criterion we accept H0 for both types of inspections. This supports the assumption that *small groups of reviewers (2-4 inspectors) find in average the same amount of defects as large groups (more then 5 inspectors).*

6 Discussion

Both assumptions stated in sections 4.4 and 4.5 were proved for *a given data set* with p=0.05. Low p-level leaves small probability that the results have random nature.

Number of weaknesses should be mentioned. First is that experimental material is lacking some background information. There is no information about reading techniques which proved to be efficient in combination with team size [5, 6] or reviewer' experience/role which is an important influencing factor [5, 9]. This was one of the reasons why we have taken preparation effort only as a time factor.

Secondly, each inspection is characterized by unique combination of author, type and quality/novelty of the artifact, time to prepare and other parameters that are often difficult to measure especially in industrial conditions. For our study we assumed that documents and code produced in the company are of similar quality, inspectors have equal experience and prepare for the meetings using same technique. We made these assumptions because of the large available amount of data. We used nonparametric method when the law of big numbers functions from more then 30 observations in a sample [10]. This law smoothes influence of single factors over a large data set.

Advantages and disadvantages of this work both come from the fact that all experimentation was done on the historical industrial data. Working with historical data leaves fewer possibilities to manipulate variables. We can disregard some information but if certain attributes were not recorded we cannot add them later on. At the same time this is real data which was not accidentally manipulated to fit our hypotheses.

Results of this study were already discussed in literature. Main contribution of this work is in highlighting for the industry areas of process improvement. Considering one factor of preparation time without any additional information with high probability level we demonstrated its importance on the results of an inspection. Also we have proved that small teams in average produce same results as large teams.

7 Summary and Further Work

Inspection is very efficient and powerful instrument to ensure high quality of the product. Inspection process has many attributes that are easy to measure and some attributes that are difficult to measure, for example people factor [9].

After studying inspection data we can conclude with three recommendations:

i. Importance of the preparation time: slower reading leads to higher defect detection.
ii. Potency of small teams allows to limit number of inspectors;
iii. Usefulness of measurements. Even if some portion of information is missing industrial metrics should be used in research.

By discussing with the company the results of this study we have seen that the data is very informative and there are tendencies which are not hidden by diversity of the data or statistical errors. We have basic ideas to plan controlled industrial experiment. For the company we will try to answer the question made by Wohlin et al. [8] asking “how many reviewers do we need to achieve a certain level of effectiveness or efficiency” and we will try to find optimal time interval based on number of attributes that are not yet measured.

Acknowledgments.

To professors Maurizio Morisio and Marco Torchiano for important comments and advices in writing the paper.

References

1. Fagan, M.: Design and Code Inspections to Reduce Errors in Program Development, IBM Systems Journal, 1976
2. Gilb, T., Graham, D.: Software Inspection, Addison-Wesley Publishing Company, 1993
3. Barnard, J., Price A.: Managing Code Inspection Information, IEEE Software, pp. 59-69, March 1994
4. Votta, L.G.: Does Every Inspection Need a Meeting?, Proc. ACM SIGSOFT '93 Symp. Foundations of Software Eng., pp. 107-114. ACM, Dec. 1993
5. Biffl, S., Gutjahr, W.: Influence of Team Size and Defect Detection Technique on Inspection Effectiveness, Proc. METRICS 2001, pp. 63-77, April 2001
6. Halling, M., Biffl, S.: Investigating the influence of software inspection process parameters on inspection meeting performance, IEE Proc.-Softw., Vol. 149, No. 5, Oct. 2002
7. Porter, A., Siy, H., Mockus, A., Votta, L.: Understanding the Sources of Variation in Software Inspections, ACM Transactions on Software Engineering and Methodology, Vol. 7, No. 1, pp. 41–79, Jan. 1998
8. Wohlin, C., Aurum, A., Petersson, H., Shull, F., Ciolkowski M.: Software Inspection Benchmarking – a Qualitative and Quantitative Comparative Opportunity, Proceedings of the Eighth Symposium on Software Metrics, pp. 118-127, 2002
9. Kelly, D., Shepard, T.: Qualitative Observations from Software Code Inspection Experiments, CASCON '02, IBM Press, 2002
10. McCall, R.: Fundamental Statistics for Psychology, N.Y., Harcont, Brace E., World, 1970

Convertibility of Function Points to COSMIC-FFP: Identification and Analysis of Functional Outliers

Jean-Marc Desharnais[1], Alain Abran[1] and Juan J. Cuadrado-Gallego[2]

[1] École de Technologie Supérieure - ETS
1100 Notre-Dame Ouest, Montréal, Canada H3C 1K3
Universidad de Alcala de Hénarès, Spain
jean-marc.desharnais@etsmtl.ca; alain.abran@etsmtl.ca; jjcg@uah.es

Abstract. COSMIC-FFP (ISO 19761) represents the second generation of functional size of the software, based on its ease of understanding and use and it is applicable to various kinds of software applications; this new method has achieved rapidly an ISO recognition as an international standard as well as market acceptance in various countries. Several organizations are therefore interested in using convertibility ratios between COSMIC- FFP and first generation of functional size measurement (in particular Function Point Analysis – FPA - ISO 20926), in order to leverage data from their historical databases of software measures. Previous convertibility studies have indicated that convertibility of FPA to COSMIC-FFP can be simple, with a very good correlation for most MIS projects, but that there are some outliers for which convertibility is less straightforward. This study analyzes a new data set of 14 projects measured with both sizing methods, and for which measurement results are available at the detailed level. The analysis reported here identifies reasons why, for some MIS projects, convertibility is not so straightforward. This analysis also provides lead indicators to identify outliers for convertibility purposes.

Keywords: ISO 19761, COSMIC-FFP, IFPUG, Convertibility, Functional Size Measurements (FSM), Software Measures.

1 Introduction

Since the late ’70s, function points have been used as a measure of software size to calculate project productivity and project estimates. Even though a large number of variants of the Function Point Analysis (FPA) method have been proposed over the years to tackle various weaknesses in the design of the original FPA method, only four have achieved recognition as ISO measurement standards:

- ISO 19761: COSMIC-FFP [ISO 03a].
- ISO 20926: Function Point Analysis (e.g. IFPUG 4.1, unadjusted function points only) [ISO 03b];
- ISO 20968: Mk II [ISO 02]
- ISO 24570: NESMA [ISO 05] (A Dutch interpretation of FPA v. 4.1 which produces similar results [NESM04]).

The FPA, MarkII and NESMA methods were primarily designed to measure business application software. COSMIC-FFP, the newest method, was designed to handle other types of software as well, such as real-time, telecommunications and infrastructure software.

Organizations interested in converting to the newest COSMIC-FFP measurement method have expressed interest in a convertibility ratio that would allow them to leverage their investments in historical data measured with FPA. The goal of this paper is to provide industry with insights into this issue of convertibility between FPA and COSMIC-FFP. All convertibility studies reported here were carried out with duplicate measurements using both COSMIC-FFP and FPA (or the NESMA equivalent) on the same set of functional user requirements (FURs). The specific versions of methods used in each convertibility study are documented for each study.

This paper is organized as follows: an overview of related work is presented in section 2, and the new data set in section 3. The convertibility formulae based on the total FPA size and FPA transactions size are presented in section 4. Outliers are identified and discussed in section 5 and a summary in section 6.

2 Related work

The following preconditions exist in all studies reported here:

- All functionalities inside the boundary of the software being measured are included in the measurement.
- Measurements have been taken from the human end-user’s viewpoint.
- FPA is considered not to include the value adjustment factor (VAF), in conformity with ISO 14143-1 [ISO98] and ISO 20926, that is, unadjusted function points (UFP).

Data from both the Fetcke 1999 study and the Vogelezang and Letherthuis 2004 study were included in the discussion on convertibility in the COSMIC Implementation Guide to ISO 19761 [ABRA03, chapter 8]. They are discussed here as individual data sets.

Fetcke 1999

In the Fetcke 1999 study [FETC99], four software applications of a data storage system were measured. These are business applications with few data entities.

The linear regression model of Fetcke’s data provides the following convertibility formula, where ‘Cfsu’ represents COSMIC-FFP functional size units and ‘UFP’ represents unadjusted function points, with a very high coefficient of determination (R^2) of 0.97:

$$\mathbf{Y(Cfsu) = 1.1 * (UFP) - 7.6 \qquad (1)}$$

Of course, because the number of data points is small (that is, only four in the data set), care must be exercised in the extrapolation of these results to larger data sets, and to data sets from different contexts. In summary, the duplicate measurement of software containing few data files and from the human end-user's viewpoint gave very similar results, and a convertibility formula with a slope fairly close to 1.

Vogelezang & Lesterhuis 2003

In the Vogelezang & Lesterhuis 2003 study [VOGE03, VOGE04], the COSMIC-FFP measurements were carried out on 11 projects already measured with the NESMA FPA (ISO 24570).

The linear regression model of this data set provides the following convertibility formula, with a coefficient of determination (R^2) of 0.99:

$$\mathbf{Y(Cfsu) = 1.2 * (UFP) - 87} \qquad \mathbf{(2)}$$

Vogelezang and Lesterhuis postulate that the constant 87 probably owes its existence to the counting of the logical files of data (ILFs and EIFs) in FPA [VOGE04], which are not directly included in COSMIC-FFP; this interpretation suggests that the high value of 87 might not be due entirely to the error term alone in this model.

With this specific data set, the two largest projects have significant influence on the regression model: it can therefore be observed that the conversion formula does not work well for small projects, those with fewer than 200 NESMA points, even providing negative numbers, which is not possible in practice. This means that, for small projects in this environment, distinct regression models should be built using only data within a relatively similar range. For instance, this data set was split into two ranges: fewer that 200 UFP and over 200 UFP. The linear regression model of the subset of data with fewer than 200 NESMA points provides the following convertibility formula, with a coefficient of determination (R^2) of 0.85 [ABRA05]:

$$\mathbf{Y(Cfsu) = 0.75 * (UFP) \quad - 2.6} \qquad \mathbf{(3)}$$

The convertibility formula from equation (3) with a slope of 0.75 and a much smaller error term of -2.6 is more relevant for representing small projects in this data set, and leads to a much smaller convertibility delta, both in absolute and in relative terms.

Next, the linear regression model of the data subset for projects larger than 200 NESMA points provides the following convertibility formula, with a coefficient of determination (R^2) of 0.99 [ABRA05]:

$$\mathbf{Y(Cfsu) = 1.2 * (UFP) - 108} \qquad \mathbf{(4)}$$

The models for the full data set and for the data set of projects over 200 NESMA points are fairly similar in terms of both their slope and their error terms. There is still, however, a large difference in convertibility results for project 8, at 260 NESMA points, both in absolute and in relative terms. This means that there must be some peculiarities in the way functionality is measured that leads to non straightforward convertibility in some instances.

Desharnais 2005 data set

The duplicate measurement results reported in [ABRA05] were collected in 2005 using FPA 4.1 and COSMIC-FP 2.2. This data set comes from a single governmental organization and was measured using the documentation of completed projects.

The linear regression model of the Desharnais 2005 data provides the following convertibility formula, with a coefficient of determination (R^2) of 0.91:

$$\mathbf{Y(Cfsu) = 0.84 * (UFP) \quad + 18} \qquad \mathbf{(5)}$$

Again, a large difference in convertibility results for one project was noted, both in absolute and in relative terms. This means, again, that there must be some peculiarities in the way functionality is measured that leads to non straightforward convertibility for this project.

In the FPA measurement method, the data are taken into account from multiple perspectives, once as logical data files (ILF – Internal Logical Files, and EIF – External Interface Files) and once again whenever there are references in FPA transactions (Input, Output, Enquiries transaction types). This has already been noted in [VOGE03a], where it is reported that, in FPA-like methods, 30 to 40% of functional size comes from the data files. By taking into account only the FPA data file points from the FPA transaction-type points, it was investigated next whether or not a better convertibility ratio could be derived by excluding the FPA data files, that is, by taking into account only the size from the transactions (UFP-TX).

With the FPA points for the transactions only and the linear regression model of the data in Figure 7, which provides the following convertibility formula with a coefficient of determination (R^2) of 0.98, we have:

$$\mathbf{Y(Cfsu) = 1.35 * (UFP\text{-}TX) + 5.5} \qquad \mathbf{(6)}$$

Thus, there is a slight improvement in the (R^2) for the convertibility formula when using only the results of the transactions for FPA, instead of the total size derived from both data and transactions; again, with such a small data set, this should be taken as indicative only, and should be investigated with larger data sets.

3 Desharnais 2006 data sets

3.1 Context

In 2006, another set of 14 MIS projects was measured using Function Point Analysis (IFPUG version 4.1) and COSMIC-FFP 2.2. The FPA and COSMIC-FFP measurements were taken concurrently using of the same documentation by a single expert in both measurement methods. All 14 projects come from a single governmental organization (different from the one reported in the Desharnais 2005 study). As for the data sets reported in the literature review, the measurements were taken from the user's viewpoint, that is, taking into account that most of the functionalities of the software involve interaction with a human, which is typical of business software applications.

This data set is presented in Table 1. While the data are available at the function-type level for FPA and at the data movement level for COSMIC-FFP, only the totals at the function-type levels are presented in Table 1.

ID.	FPA size (in Function Points – FP)						COSMIC-FFP size (in Cfsu)				
	Input	Output	Inquiries	ILF and EIF	Total FP		Entry	Exit	Read	Write	Total Cfsu
1	31	145	95	112	383		63	155	120	26	364
2	98	162	168	217	647		96	233	91	45	565
3	104	127	71	98	400		59	125	146	68	398
4	64	55	25	61	205		39	66	55	28	188
5	94	135	66	77	372		52	158	173	65	448
6	22	29	22	53	126		20	37	24	7	88
7	24	21	10	56	111		11	41	47	16	115
8	94	51	72	70	287		45	103	104	46	298
9	202	54	148	96	500		78	110	198	193	579
10	83	128	28	105	344		54	114	92	31	291
11	55	88	69	105	317		49	119	98	28	294
12	103	49	57	49	258		50	86	78	38	252
13	42	35	10	26	113		19	23	39	33	114
14	157	115	70	105	447		67	149	167	84	467

Table 1: Total functional size at the function-type level

3.2 Distribution of functional size at the transactional and data movement levels

Analysis of the distribution of the transactions in FPA and the distribution of data movements in COSMIC-FFP is one way to identify discrepancies in the measurement results. Figure 1 presents the distribution of function-type sizes for the FPA measurement results: the total sizes of FPA Input and Output function types have the same percentage of 36%, while the Inquiry function type is lower at 28% (Figure 1). The distribution of function-type size for this set of 14 projects is reasonably comparable to the distribution of function types of the 3,161 IFPUG projects in the February 2006 edition of the International Software Benchmarking Standards Group – ISBSG – repository (Figure 2). Figure 3 presents the distribution of data movement types in COSMIC-FFP units for this data set of 14 projects: the eXits have the greatest ratio at 43%, and the Writes the lowest at 7%.

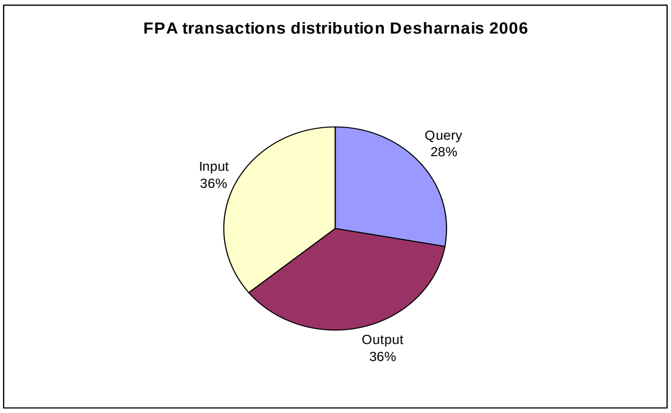


Figure 1: FPA transaction size distribution (N= 14)

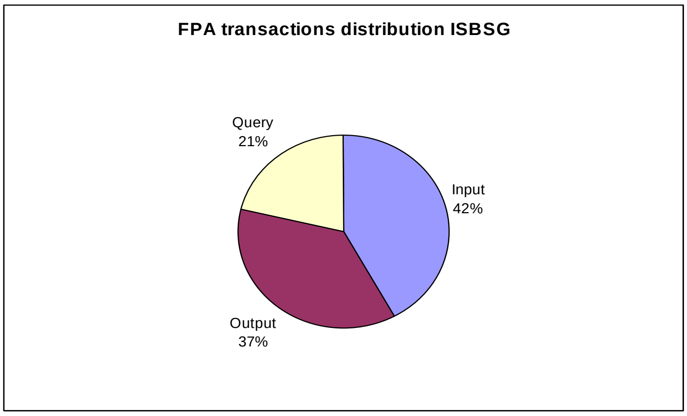


Figure 2: ISBSG transaction size distribution (N=3,161)

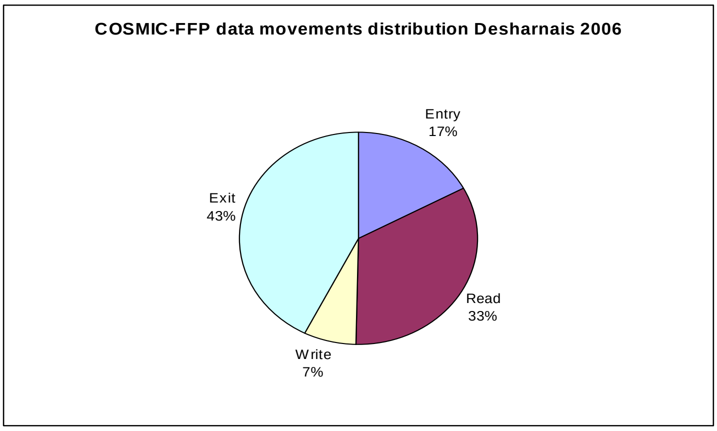


Figure 3: COSMIC-FFP data movement distribution (14 projects)

3.3 Quality of the documentation

It is important to stress that the quality of the measurement results depends on the competence of the measurer, but this is not the only major factor impacting their quality. Also important is the quality of the documentation, which is the key input to the measurement process, since it has a significant impact on the measurement results, in terms of the proper number of functions identified and the assignment of the correct size to each of these functions.

For the measurement results reported here, the measurement process was performed on software projects which had been completed and were currently in use in the organization. While on the one hand there was no uncertainty in terms of whether or not there were unspecified requirements – a situation typical when measuring early in the development life cycle, on the other hand, the documentation of projects implemented cannot be assumed to be perfect, documentation often being neglected in projects, which in turn leaves software maintainers with poor documentation to work with.

While the functional size of the 14 software applications implemented was measured from the maintenance documentation, the quality of the documentation was classified using the following criteria:

A: Full documentation – that is, all the required detailed information was available for the measurement process;

B: Documentation, but with an incomplete data model;

C: All functions identified;

D: Only the number of functions identified, but without enough detail for accurate measurement;

E: Implicit documentation of a function.

Using the above criteria, the following observations were made:

- For 12 projects, the documentation was of good quality (= A or B) for over 95% of their functional processes.
- For projects 3 and 9, the documentation was of good quality for only 62% and 71% of their functional processes respectively.

On a weighted average, the project documentation was considered to be of good quality for more than 90% of the functions measured.

For the projects with documentation of less than good quality, assumptions had to be made for the measurement of the functions not documented well enough for precise measurement; however, since the same assumptions were made for the concurrent duplicate measurement with both methods, this could not be a source of significant distortions for convertibility purposes in this specific case study reported here r.

4 Identification and analysis of outliers

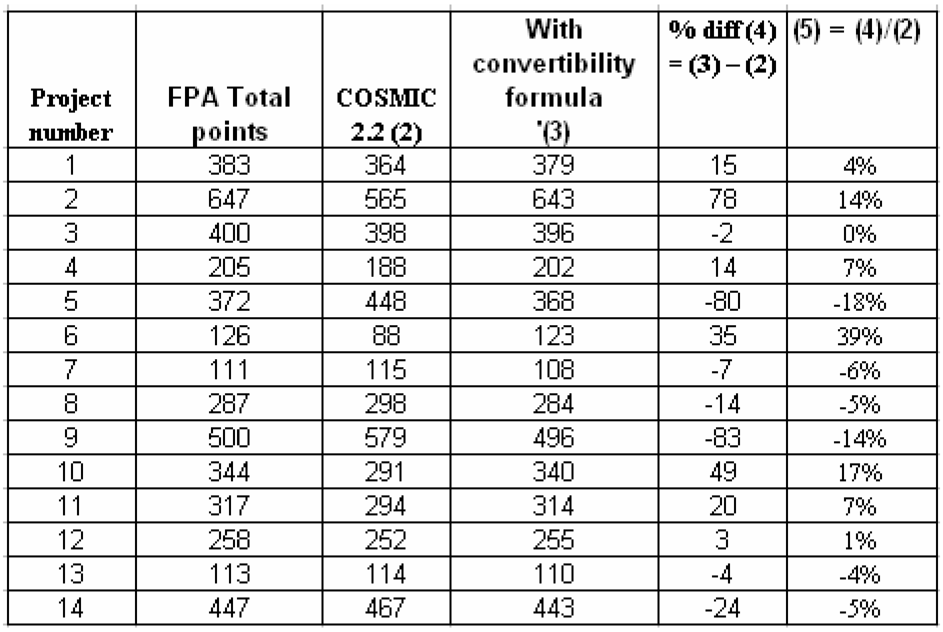
4.1 Convertibility analysis using total FPA size

The first convertibility analysis investigates the FPA to COSMIC-FFP relationship based only on total FPA size, that is, on the summation of all FPA size units, without looking into the details of the measurement. The measurement results of the duplicate measurement of the 14 applications are reported in Table 2 and are presented graphically in Figure 4, with the FPA data on the x-axis and the COSMIC data on the y-axis.

The linear regression model of the data in Figure 4 provides the following convertibility formula, with a coefficient of determination (R^2) of 0.93:

$$Y(Cfsu) = 1.0* (UFP) \quad - 3 \qquad (7)$$

This convertibility formula represents an almost 1 to 1 convertibility ratio. This does not mean, however, that this produces entirely accurate results, as can be seen in column (4) of Table 2 (e.g. the difference between the convertibility results and the direct measurements in COSMIC-FFP). Column (5) presents the same difference as a percentage: it can be observed that for 9 projects out of 14 the relative difference is less than 10%, for 4 projects it is between 10 and 20% and for one project it is 39%. The weighted average of the absolute difference (from column (4)) for all projects is (absolute (difference) / total COSMIC size) = 428 Cfsu / 4,461 Cfsu = 9.5%.



Project number	FPA Total points	COSMIC 2.2 (2)	With convertibility formula '(3)	% diff (4) = (3) – (2)	(5) = (4)/(2)
1	383	364	379	15	4%
2	647	565	643	78	14%
3	400	398	396	-2	0%
4	205	188	202	14	7%
5	372	448	368	-80	-18%
6	126	88	123	35	39%
7	111	115	108	-7	-6%
8	287	298	284	-14	-5%
9	500	579	496	-83	-14%
10	344	291	340	49	17%
11	317	294	314	20	7%
12	258	252	255	3	1%
13	113	114	110	-4	-4%
14	447	467	443	-24	-5%

Table 2 Convertibility comparison on Total FPA Size (N= 14)

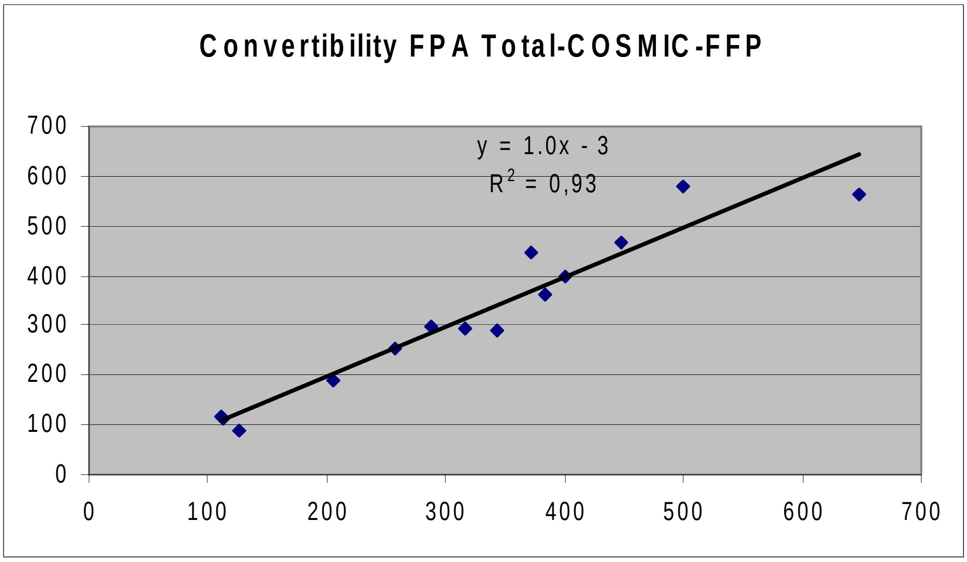


Figure 4: Convertibility model on Total FPA Size (N=14)

4.2 Convertibility analysis based on the FPA transactions sizes

The next convertibility analysis is based on the sizes of the three FPA transaction types (Input, Output, Inquiries) – that is, excluding the sizes from the Internal and External logical files of the IFPUG method.

The measurement results of the duplicate measurement of the 14 applications are reported in Table 3 and presented graphically in Figure 5, with the FPA transaction sizes on the x-axis and the COSMIC data on the y-axis.

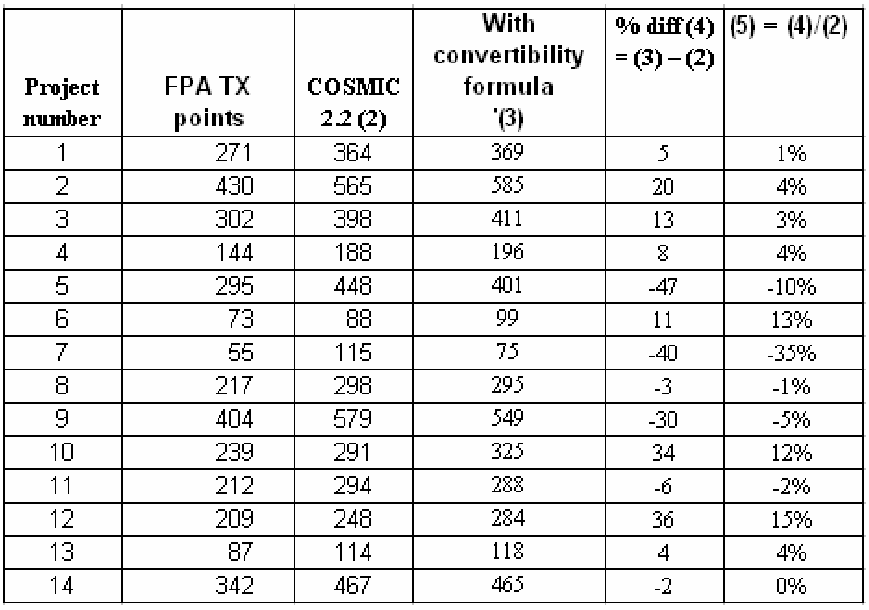
The linear regression model of the data in Figure 5 provides the following convertibility formula, with a coefficient of determination (R^2) of 0.98:

$$Y(Cfsu) = 1.36^* (UFP\text{-}TX) \quad + 0 \qquad (8)$$

This convertibility formula is, of course, different from (7), since it is derived from a distinct basis (FPA transaction size rather than total FPA size) and its R^2 (0.98) is slightly better than the previous one (0.93). Again, this does not mean that entirely accurate results are produced for all projects converted, as can be seen in column (4) of Table 3, which represents the difference between the results of the convertibility and direct measurements in COSMIC-FFP. In column (5), the same difference is presented as a percentage. Some significant improvements can be observed:

- 9 projects out of 14 have a very small relative difference of less than 5%;
- 4 projects have a relative difference of between 10% and 15%; and
- 1 project has a relative difference of 35%.

The weighted average of the absolute difference for all projects is (absolute (difference) / total COSMIC size) = 259 Cfsu / 4,461 Cfsu = 5.8%; this is a major improvement when compared to the 9.5% difference with convertibility based on total FPA size..



Project number	FPA TX points	COSMIC 2.2 (2)	With convertibility formula '(3)	% diff (4) = (3) – (2)	(5) = (4)/(2)
1	271	364	369	5	1%
2	430	565	585	20	4%
3	302	398	411	13	3%
4	144	188	196	8	4%
5	295	448	401	-47	-10%
6	73	88	99	11	13%
7	55	115	75	-40	-35%
8	217	298	295	-3	-1%
9	404	579	549	-30	-5%
10	239	291	325	34	12%
11	212	294	288	-6	-2%
12	209	248	284	36	15%
13	87	114	118	4	4%
14	342	467	465	-2	0%

Table 3 : Convertibility comparison on FPA transaction size only (N=14)

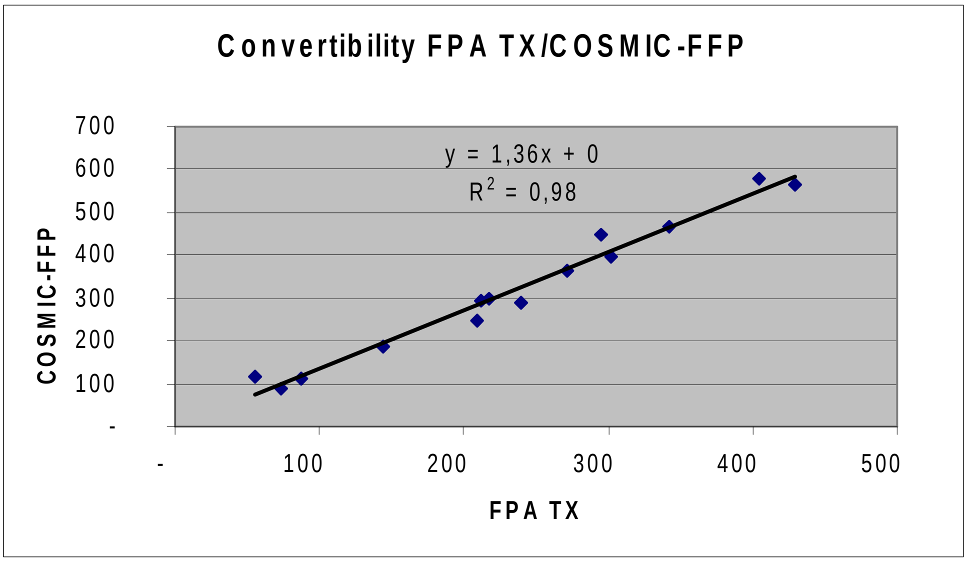


Figure 5: Convertibility model on FPA transaction size only (N=14)

o

5 Discussion

5.1 Analysis at the FPA transaction level

While measuring the 14 MIS projects in 2006, it was observed, in the context of the documentation available for measuring these projects, that the concept of functional process in COSMIC-FFP is, in practice, equivalent to the concept of functional transaction type in FPA. When identifying a COSMIC-FFP functional process in the documentation of a project, it was observed that there is an equivalent FPA elementary process.

While the mapping of the total size at the total level is reasonably direct from FPA to COSMIC-FFP for this sample, there are differences at the lower levels (sub-totals at the function-type level and individual measurement results at the functional process level).

As noted in previous studies [VOGE03a] and [ABRA05], there is a difference in measuring the size of the COSMIC-FFP functional processes when compared to FPA transactions: the COSMIC-FFP functional process is based on the number of data movements in a functional process, while the FPA transactions (Input, Output and Inquiries) are based on the number of Data Element Types (DET), File Type References (FTR) and on sets of weights (from 3 to 7) provided in different FPA weight tables. In FPA, the sizes of the files (ILF and EIF) are added to the sizes of the transactions (input, output, query) to obtain the FPA total size in UFP; in FPA, this is equivalent to adding together distinct entity types (eg. adding the sizes of Tables to the sizes of TV sets) leading to totals without a clear interpretation of the summation results and, thereby, leaving end results difficult to interpret from a size viewpoint. In COSMIC-FFP, there is no such equivalent of adding something of a different nature: only data movement types – Entry, Exit, Read and Write – are added together; it is therefore expected that convertibility at the transaction level be more meaningful.

In addition, when analyzing the results for each functional process and each transaction, it was observed that the COSMIC-FFP sizes of all the functional processes are, in this data set, systematically equal to or higher than the FPA size of corresponding transactions. There are at least two explanations for this:

- The weights in the FPA table for the transactions are limited to 7 points, while there is no such upper limit in the number of data movements in a COSMIC functional process.
- The measurement of the error messages: on the one hand, Function Point Analysis version 4.1 rules includes error messages as a part of the transaction without assigning it additional points, while, on the other hand, COSMIC-FFP recognizes an error message as one additional data movement in a functional process. The result is that a simple functional process has one more COSMIC-FFP size unit (e.g. the data movement that takes into account the error message in the measurement process).

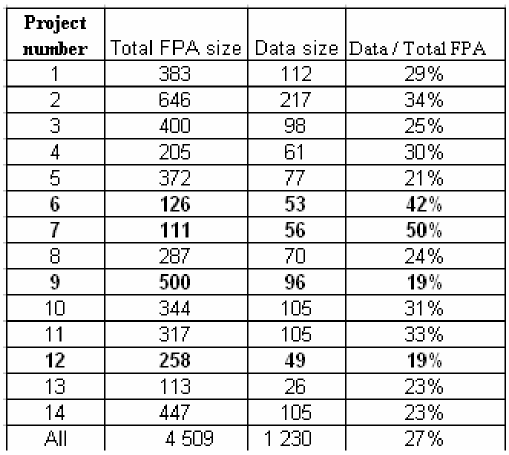
5.2 Analysis of variations at the FPA data level and identification of functional outliers

In the previous section, it was seen that, for this data set, the COSMIC-FFP size of an FPA transaction was systematically larger, while the FPA data size was not taken into account.

The ratio of FPA Data size to FPA Transaction size was analyzed next. In actual measurements with FPA, the ratio of files to transactions is not constant and can vary across projects. The potential impact of this is analyzed next: Table 4 presents the ratio of FPA size allocated to the data files, referred to as Data points in Table 4. In this data set, the average ratio of FPA data file size over the PFA total size is 27% (bottom line – Table 4); this ratio varies from a low of 19% to a maximum of 50%.

For this analysis, the four projects further away from the average on functional distribution (identified in bold in table 4) are investigated, that is, the 2 projects with the lowest ratio (of 19%) of data file points (projects 9 and 12) and the 2 with the highest ratios of 42% and 50% (projects 6 and 7). For these 4 projects, we look at the relative error of convertibility based on FPA transaction size in Table 3:

- Projects within the lower ratios: project 9 has a relative convertibility error of -5% (Table 3), while project 12 has a convertibility error of 15%.;
- Projects within the highest ratios: project 6 has a relative convertibility error of 13%, while project 7 has by far the highest convertibility error, at 35%.



Project number	Total FPA size	Data size	Data / Total FPA
1	383	112	29%
2	646	217	34%
3	400	98	25%
4	205	61	30%
5	372	77	21%
6	**126**	**53**	**42%**
7	**111**	**56**	**50%**
8	287	70	24%
9	**500**	**96**	**19%**
10	344	105	31%
11	317	105	33%
12	**258**	**49**	**19%**
13	113	26	23%
14	447	105	23%
All	4 509	1 230	27%

Table 4 : Ratio of FPA data file size over FPA total size (N=14 projects)

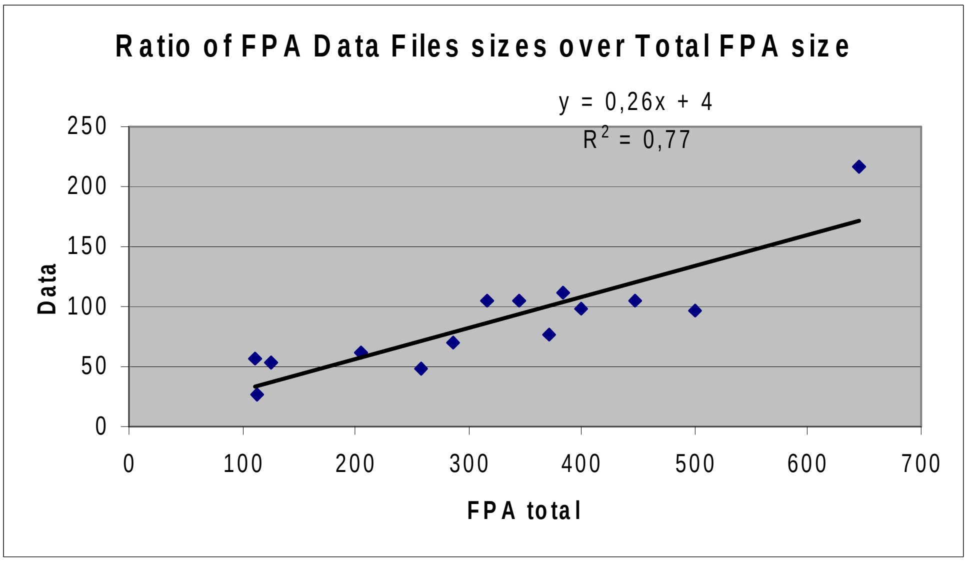
While outliers on one characteristic can help identify candidates for explaining variances on another characteristic, this does not necessarily constitute a cause-and-effect relationship. Outliers are, however, good candidates for identifying hypotheses for further research. In particular, project 7, with both the largest variation of data sizes from the average and the largest relative convertibility delta (e.g. 35% – Table 3) needs to be investigated at a detailed level. Similarly, projects 6, 7 and 9 should be looked into at the detailed level.

From Table 2, one project (6) shows a 39% convertibility error when comparing the total FPA points (126) and converted COSMIC-FFP points (88), while two others projects of similar size (project 7 with 111 FPA and project 13 with 113 FPA) have only a 6% and -4% difference with the convertibility formula (7). In terms of the ratio of FPA data file size to transaction size, this project (6), has its FPA data file size representing 42% of the total (Table 4), a long way from the average of 27%. This, then, contrasts with project 7, with a 50% data file ratio (table 4) and only a -6% convertibility error (table 2) on total FPA size; by contrast, this variation for project 7 is -35% (table 3) when convertibility is derived from FPA transaction size. Such variation from average functional profiles and from basis of convertibility (total FPA size or FPA transaction size) can provide clues for identification of convertibility outliers, but does no provide a full rationale (that is, it does not lead necessarily to a cause-effect relationship for convertibility purposes).

To provide an explanation for this, we need to look at project 6 in Table 1 to see that the number of eXits and Reads are relatively high, and this increases the number of points. When looking more closely (eg. at the detailed duplicate measurements results of each functional process) of project 6, we observed that out of its 11 functional processes, 4 have a high number of Reads (8 data movements) while 1 functional process has a high number of Writes (8 data movements). This means that in COSMIC-FFP the number of data movements is higher, since a higher number of files (e.g. data groups) are handled by both Reads and Writes. This is not so in the case for project 7 which does not have a higher number of FPA functional processes over the FPA limits set by its weights tables, but still has the lowest ratio of FPA transaction size (50%) with respect to total FPA size.

For these 14 projects, 688 functional processes are identified, and we did a survey of the number of data movements for each functional process: 17 (2%) functional processes have more than 4 data movements for a Read, 30 (4%) data movements for a Write, 525 (76%) have 1 or 2 data movements for a Read and 439 (64%) functional processes have 1 or 2 data movements for a Write. Project 6 is atypical because 4 of its 11 functional processes have more than 4 data movements (36%). FPA, with its table upper limit of 6 or 7 points, cannot correspondingly provide larger size for those larger functional processes.

-



6 Summary

This 2006 study is a replication, but with a larger data set (14 projects from a single organization), of the 2005 study (6 projects from a single, but different organization), and the findings were similar at the total FPA size and FPA transaction size. They were also comparable to the findings of previous studies: the convertibility formula is within the same range and work for most of the software measured with both methods: however, there are a few projects for which the convertibility formula provides poor results.

In summary, this analysis together with the findings highlighted in the literature review, indicates that a relatively reasonable convertibility formula can be obtained for each MIS data set, but that there are some variations in the convertibility formulae across organizations. These variations could be caused by extraneous factors, such as non homogeneity in the distribution of FPA function types (ratio of data files to transactions sizes) and the documentation types (varying across development methodologies) or variation in the quality of such documentation, across the organizations where the measurements were derived.

These analyses also provided an indication that convertibility can be straightforward either based on total FPA size or FPA transaction size for the majority of the projects in a data set, even though there are larger variations for a few projects. These analyses also indicated that for this data set the convertibility error is smaller (e.g. weighted average difference of 5.8%) when the basis for convertibility is based on FPA transaction size only. This means that convertibility of a full portfolio of software applications could be reasonably accurate overall, but that a few individual projects would show some larger variation from the values predicted by the convertibility formulae. This study has also identified some candidate lead indicators to explain these greater variations, such as a large dispersion of FPA file ratio from the sample average, as well as the ratio of FPA processes the size of which is constrained by the upper limits in the FPA tables of points.

This study did not investigate more complex contexts, such as those having projects with more algorithmic-rich processes and/or when there are software users other than software or engineered devices, such as in the case of real-time software. Under these latter conditions, of course, backward convertibility (from COSMIC-FFP to FPA) is not of much interest, nor is it an issue, since functionality related to non-human users (such as interactions with sensors or controllers in embedded software, or in multi-layered software) is not usually taken into account and not measured in first generation measurement methods.

References

[ABRA97] Abran, A.; Maya, M.; Desharnais, J.-M.; St-Pierre, D., *Adapting Function Points to Real-Time Software*, in American Programmer, vol. 10 , 1997, pp. 32-43

[ABRA99] Abran, A.; Desharnais, J.-M.; Oligny, S., *Measuring with Full Function Points*, in ESCOM SCOPE 99, Herstmonceux Castle, England , 1999, pp. 104

[ABRA00] Abran, A.; Symons, C.; Desharnais, J.-M.; Fagg, P.; Morris, P.; Oligny, S.; Onvlee, J.; Meli, R.; Nevalainen, R.; Rule, G.; St-Pierre, D., *COSMIC FFP Field Trials Aims, Progress and Interim Findings, 11th European Software Control and Metric Conference (ESCOM SCOPE 2000)*, Munich, Germany, 2000, pp. 31.

[ABRA03] Abran, A., Desharnais, J.-M., Oligny, S., St-Pierre, D., Symons, C., *Measurement Manual COSMIC Full Function Points 2.2 - The COSMIC Implementation Guide for ISO/IEC 19761*, École de technologie supérieure, Université du Québec, Montréal, Canada, 2003 url: www.gelog.etsmtl.ca/cosmic-ffp

[ABRA05] Abran, A., Desharnais, J.-M., Azziz, F., “ Measurement Convertibility: From Function Points to COSMIC-FFP,” 15th International Workshop on Software Measurement – IWSM 2005, Montréal (Canada), Shaker Verlag, Sept. 12-14, 2005, pp. 227-240.

[DEKK03] Dekkers, T., Vogelezang, F., *COSMIC Full Function Points: Additional to or replacing FPA*, Sogeti Nederland B.V., 2003.

[DESH00] Desharnais, J.-M.; Abran, A.; St-Pierre, D. , *Mesure de la taille fonctionnelle des logiciels temps réel*, in Revue Génie Logiciel, Paris, France, vol. 54, 2000, pp. 8-13 .

[FETC99] Fetcke, T., *The warehouse software portfolio, a case study in functional size measurement*, Technical report no. 1999-20, Département d'informatique, Université du Quebec à Montréal, Canada, 1999

[Ho 99] Ho, V.T.; Abran, A.; Fetcke, T., *A Comparative Study Case of COSMIC-FFP, Full Function Point and IFPUG Methods,*Département d'informatique, Université du Québec à Montréal, Canada, 1999.

[ISO98] ISO/IEC 14143-1:1998 Information technology -- Software measurement -- Functional size measurement -- Part 1: Definition of concepts, International Organization for Standardization, Geneva, 1998.

[ISO02] ISO/IEC 20968: 2002, Software engineering -- Mk II Function Point Analysis -- Counting Practices Manual, International Organization for Standardization -- ISO, Geneva, 2002.

[ISO03a] ISO/IEC19761:2003, Software Engineering -- COSMIC-FFP -- A Functional Size Measurement Method, International Organization for Standardization - ISO, Geneva, 2003.

[ISO03b] ISO/IEC 20926: 2003, Software engineering -- IFPUG 4.1 Unadjusted functional size measurement method -- Counting Practices Manual, International Organization for Standardization -- ISO, Geneva, 2003.

[ISO05] ISO/IEC 24750: 2005, Software engineering -- NESMA functional size measurement method version 2.1 -- Definitions and counting guidelines for the application of Function Points Analysis, International Organization for Standardization -- ISO, Geneva, 2005.

[MORR98] Morris, P.; Desharnais, J.-M., *Measuring ALL the Software not Just what the Business Uses*, in IFPUG Fall Conference, Orlando, Florida, 1998.

[NESM04] NESMA, Netherlands Software Measurement Association web site 2004, *http://www.nesma.nl/english/nesma&ifpug.htm#bm_Similarities_between_NESMA_and_IFPUG*

[VOGE03]Vogelezang, F., Lesterhuis, A., *Applicability of COSMIC Full Function Points in an administrative environment: Experiences of an early dopter,* 13th International Workshop on Software Measurement – IWSM2003, Shaker Verlag, Montréal, 2003.

[VOGE04] Vogelezang, F., *Implementing COSMIC-FFP as a replacement for FPA*, 14th International Workshop on Software Measurement – IWSM-Metrikon 20043, Konig-Winsterhausen, Germany, 2004.

Getting the right ‘scale’ in tool based structural fitness measurement

John Kuriakose Mudit Kaushik

SET Labs, Infosys Technologies
Pune, 411057, India
john_kuriakose@infosys.com mudit_kaushik@infosys.com

Abstract. How do we ensure a predictable level of internal code quality that reconciles with the current scale and context of software engineering characterized by very large applications, geographically dispersed teams and tight deadlines? We do not have enough experts to scale our code review method to the practical context of software development.

We introduce structural fitness as one aspect of software quality that measures effective use of a language features and a well defined posture in terms of software metrics. Our measurement model defines the concept of a Flavour as the key abstraction to formally decompose a Quality Goal into measurable metrics and code rules. The model is then interpreted by a tool during the actual measurement phase to aggregate low level measurements to high level goals and thus provides a reasonable measure of structural fitness. The measure accumulates the violations and is based on the observation that it is easier to measure badness than it is to measure goodness in software.

This precise formal model thus reduces the amount of subjective interpretation required for metrics related measurements and also enables suitable assessment and comparison of large application portfolios.

Introduction

There is a dearth of experts in Software Development today and yet the scale of our delivery in Software engineering is increasing accompanied by customer expectation of more predictability, better visibility and lowering costs. Development teams are being stretched to deliver more tested, integrated lines-of-Code within a fixed time frame. This context has stretched some of our current methods including those related to software quality assessment. Specifically, we see holes in the current process for scalable and precise quality assessment of code.

We have chosen to focus on one aspect of software quality - the internal code quality that we identify as *Structural Fitness*. We understand that quality is as much about Functional fitness or alignment to stated requirements as it is about internal quality. However, our choice to focus on Structural Fitness is based on certain observations in the practical software engineering context.

- There is a healthy awareness and mature tools and method for demonstrating functional fitness by testing in the industry.

- Large projects typically fail to meet customer expectation in code quality even while delivering functionally fit systems.
- The dearth in experts to match the scale of our software delivery affects the predictability of our structural fitness assessment and management.

It is simply not feasible for a human expert to review code to ascertain its internal quality. The issue is not that they do not know how to measure metrics and physical attributes of software. The issue is of proper interpretation guidance on the measured results. Project teams are not able to interpret the results and drive refactoring decisions based on them.

Problem Context

- How do scale up software quality assessment related to structural fitness to match the current scale of software development?

- How do we reduce the amount of subjective interpretation required for tool based software measurement?

Measuring Software Quality is about measuring the goodness in software. Therefore our objective is about finding precise ways to measure goodness characteristics in software related to internal code quality. IEEE 9126 has a well define taxonomy of software quality attributes that represent these goodness facets. The facts that are within our objective are maintainability, Portability and Usability.

Related work

There is substantial previous work that has contributed to the definition [ChK1994, MensLz2002,] and validation [BMB1999, ZuseBl1989, Fenton1991] of software measurement and metrics. Further, many models have been espoused to align quality goals like maintainability to measured attributes in software. These models rely on various mathematical and statistical models to derive the correlation between desired quality goals and metrics. [ColemanEtal1994, SimonEtal2001, MeyerNiere].

Previous work by [BasiliWs1984, Basili1992] related to the Goal–Question–Metric (GQM) paradigm has already laid out a well defined framework for incorporating measurement into any improvement program. GQM certainly applies to Software metrics but we observe that tools have not incorporated features that enable the end-to-end correlation from Measurement Goals to measured values.

Further, as recently observed [Marinescu2002] there is lack of clear interpretation guidance as how the results from measurements should be used to drive actions (refactoring) or decisions within the software engineering context. The importance of proper correlation can be understood by the fact that *measurement only generates* *'data' not the 'information'*. And for transforming data into information we need to interpret it well.

Approach

Our previous work [JohnMudit2006] introduced the motivation and concepts underlying our Goal-Flavour Analysis (GFA) approach to software measurements and how it and addresses some of the outstanding issues in applying measurement in large scale software projects. We have mentioned how GFA is a refinement of the earlier Goal-Metric-question proposed earlier [Basili1984, Basili1992].

The GQM paradigm consists of three steps:

- Generate a set of *goals* based upon the needs of the organization.
- Derive a set of *questions*.
- Develop a set of *metrics* which provide the information needed to answer the questions.

The GFV refines the GQM in the context of tool based measurement and analysis by introducing the concept of a **Flavour** as the key abstraction to formally decompose a goal to metrics. A Flavour is represented in our model as a directed acyclic graph and defines how exactly a goal is decomposed and expressed in terms of metrics.

This model is then interpreted during the measurement phase when the metric measurements and code rules produce Violations based on threshold settings and these violations are used to derive measured values for the parent goal node in the graph.

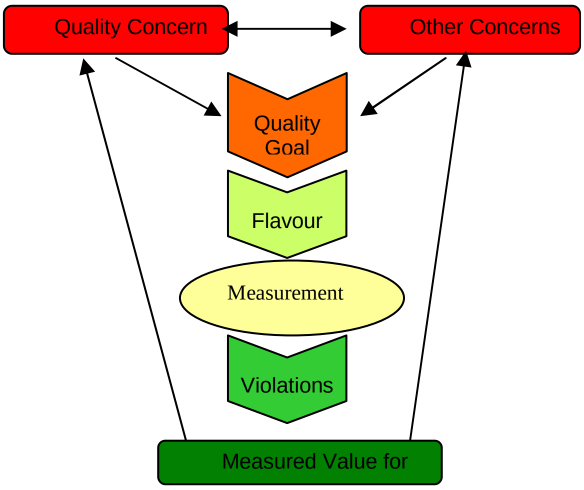


Fig 1: Goal-Flavour-Violations (GFV) process

The Code structure and hierarchy

Each programming language has a grammar that defines the basic and composed tokens used in composing statements in that language. The grammar rules defines a containment hierarchy in java that start with Modules (packages) and then to Types (Classes and Interfaces) and then to Features (Attributes and Methods).

Package→ Type → Feature represents the code elements in Java and their hierarchy of containment. We have added two more virtual elements called Software System and Module (sub-system) that represent two higher units of organization to match the scale of large software systems where even packages are the not the right unit of expression to express and understand system functionality and layout.

Additionally, we also have usage based structure like the call graph that is based on the call of methods from method bodies. These structure maps well to higher level constructs like use-cases and therefore help in making measurements at use-case level.

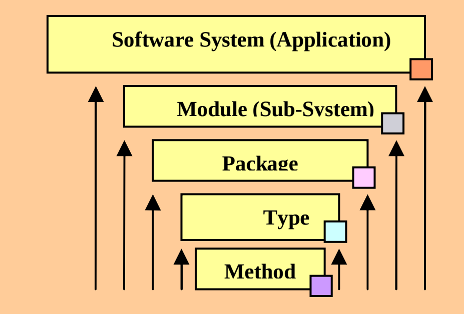


Fig. 2: Code-element hierarchy

Each metric is clearly applicable to any of the above code elements and may be accumulated to its parents by either a simple count by addition, or an Average operation or even a distinct element count operation (cardinality after set union)

Metrics and Code rules

Software metrics in our observation alone are insufficient to represent the wide variance in code quality expectation across real projects. We use Code rules in addition to metrics to define a Flavour. Code rules are the formal code patterns that reflect the mature knowledge related to use of language features like those defined in [Bloch2001]. They may also be used to encode domain specific or project specific rules related to code structure. We have found that a proper selection of metrics and code rules is sufficient to cover all real project requirements we have encountered.

We have defined 4 categories to organize our basic metric measurements. Each metric is applicable only for a particular code element in the hierarchy.

Direct metrics

These are software product metrics that have defined and established in previous works.

Size Related measurements

- Method Metrics: Lines-of-Code (LOC), Executable-Lines-of-Code (ExecLOC)
- Class Metrics: Number-of-Added-Methods (NOAM), Number-of-Inherited-Methods (NOIM), Number-of-Operations (NOO=NOAM+NOIM).
- Package metrics: Number-of-Classes (NOC), Number-of-Interfaces (NOI), Number–of-Abstract-Classes (NOAC)

Structure Related Measurement

- Class Metrics: Depth-of-Inheritance (DOI) and Number-of-Sub-Classes (NOSC)

Dependency & Coupling related measurements

- Class Metrics: Coupling-between-Objects (CBO)
- Outbound and In-bound Dependency Method Metrics: Intra-class-feature Dependencies (ICF), Intra-Package Dependencies (IPF), External Package feature dependencies (EPF)

Complexity Related Measurements

- Method Metrics: Cyclomatic Complexity (CC), Response-for-Class (RFC)
- Class Metrics: Weighted-Methods-per-Class (WMC)

Promoted Metrics

In order to provide end-user with information at the right level of detail it is necessary to define derived metrics from these basic definitions [MensLz2002]. We will refer to these henceforth as promoted metrics and they are obtained by applying a simple mathematical operation on metric measurements at code elements lower in the hierarchy.

For example: The outbound external package feature dependencies (EPF) defined at method level is based on the set of features (attribute access and method call) that are ‘used’ within a given method (M). This metrics may be promoted to the class level by simply applying the set union of all such EPF sets for all methods in a class. The resulting metric is the EPF set at Class Level. Similarly we use other promoted metrics like Total LOC (Commented or Executable) and Total Number of Classes at package level from other lower measurements. Currently we have identified three operations for promotion.

- Total Count by Addition (Sum)
- Promotion by Average (Avg)
- Total Count of Unique Elements (Unq)

The GFV Process

The Goal Flavour Violation approach that we propose for code quality assessments is based on some basic assumptions and observations.

Traditionally software quality has been understood to be about ‘goodness’ of software and quality assessment is therefore a measure of goodness in software. Our observation is that it is far easier to detect and measure bad code that it is to measure the property of goodness in code. This observation is important from our principal goal of measurements driving improvements. In order to prioritize action for improving code quality it is essential to know how far our code is from some acceptable point. We define the concepts of **Violation distance** in order to quantify code badness. The distance of violation is based on the interval scale and is obtained for those metrics and code rules for which a threshold value is set.

Violation Distance = Measured Value / Threshold Value

Certain code rules like ‘Refer to object by their Interfaces’ [Bloch2001] cannot be attributed with a threshold value. In these cases we use a simple **Violation Count** instead of the Violation distance. This violation based measurement resolves many problems mentioned by [ZuseBl1989] and [Fenton1991] related incompatible scales of measurement being compared and combined.

Measurements are useful only when it provides useful information for decision-making or leads to further action. Comparing the target and actual quality profile of an application represents an opportunity for quality measurement since it provides objective results for that comparison. Providing objective guidance for refactoring is an example of another opportunity where measurements provide great value. [SimonEtal2001]

Configuring the GFV Measurement Model

This step addresses the question - How do we formally decompose a desired quality goal into measurable units? The model uses a directed graph where the root node represents a high level quality goal and the child nodes represent sub-goals that are further refined to low-level metrics and code rules. This decomposition is essentially a top-down activity that is to be done once for a project. This step aligns the quality goals to the actual context of software development based on parameters that affect the quality profile. These parameters are

- The intrinsic complexity of the domain [Brooks1987]
- the applications contributions to business functions – its business critical nature
- and the implementation complexity of the software system based on the language, platform and other design decisions made by the development team.

The steps involved in configuring the measurement model are as follows

1. Goal(s) definition – The high level goal represents the overall intention in making measurements. For example: Java Code Quality or Maintainable Code.
2. Goal decomposition– In this step the high level goal is refined in terms of software quality attributes (G1) that refer to the degree of the goodness of the code in terms of portability (G1.c1) , code clarity (G1.c2) and maintainability (G1.c3) .Thus our goal G1 can be decomposed into smaller or sub goals G1.c1 , G2.c2 and G3.c3.
3. Concretization – Low level details are now attached to the model in this step. Both Direct and Composite Metrics along with Code Rules constitute low-level details. Code rules describe proper usage of language features and either encodes a recommended pattern or even an anti-pattern that must be avoided.
4. Setting measurement bounds – in this step we define heuristic bounds (thresholds) for metrics and applicable rules. These bounds are the threshold value that restricts the infinite space of measure values. These values are established with based on past experience, co-relation and analysis.

Rules that govern the decomposition
Comparable Sibling Rule: All immediate children from any node in the measurement model must be applicable to the same Code Element in the hierarchy.

This implies then that any decomposition that violates this rule will have to transform by adding a sub-tree as shown in figures 2 and 3.

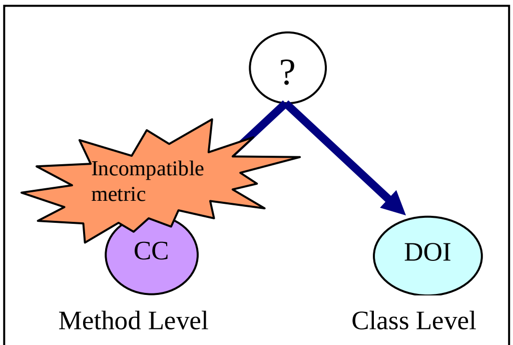


Fig 2 Comparable Sibling

For example: In figure 2 a goal has been decomposed into two metrics - Cyclomatic Complexity (CC), which is Method level Metrics and Depth of Inheritance (DOI) which is a Class level metric.

The graph has to be transformed based on this rule as shown in figure 3

The transformation involves replacing the CC metric at Method-level with the corresponding promoted metric at class level. For Cyclomatic Complexity there is a metric known as Weighted Method per Class (WMC) defined by [ChK1994] that is a ratio scale. We defined a simple aggregate called Total Cyclomatic Complexity in Class (TCC) that is a sum of the CC for all methods in a class. This definition and relation of TCC to the method level metric CC may be coded into the tool configuration or even defined by the end-user while configuring the measurement model.

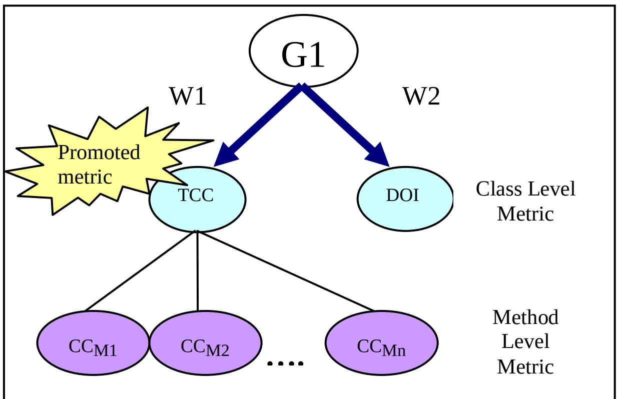


Fig 3: Transformed Graph from Fig 2

Uniform Code Element Rule: This rule follows from the Comparable Sibling Rule and states that for any sub-goal or derived metric expressed in the model the code element applicability of a parent node is the code element of its direct children.

Based on this rule the goal G1 in figure 3 is now a measurement at class level and therefore the comparable sibling rule applies at the parent of node G1.

Once the measurement model has been thus defined this actual measurement phase of GFV may be initiated.

Deriving Goal measurements from the model

The second question is how do we accumulate or aggregate low-level measurements into high level goals based on the decomposed model so as derive precise numbers for goals. This essentially is a bottom-up activity during actual measurement when goal values have to be calculated from proper expressions involving low-level measurements.

There are basically two kinds of edge relationships in the GFV measurement model. Accumulated measurements when the parent node in the graph refers to an accumulated metric calculated from metrics at a lower position in the code element hierarchy. The other kinds of edges are violation contributions that simply contribute to the total degree of badness at the parent level by contributing a measured Violation Distance (or count).

Those edges that contribute to Violations may be attached with a weight to restrict their contribution to the parent measured value. The default weight (w) is 1 and permissible values are $0 < w < 1$.

For example in figure 3 the model is now instantiated for every class in the software system since G1 is now a class level measurement. The TCC is first calculated from individual method CC measurements and then the TCC and DOI violation distance is added to obtain the Goal measure (assuming default weight =1).

The Table below shows sample measures for 4 classes in a system based on Figure 3.

Class	Number of Methods	Total Cyclomatic Complexity (TCC) Threshold=10		DOI Threshold=2		Goal measure
		Measured Value	Violation Distance	Measured Value	Violation Distance	
C1	3	30	3.0	2	0	3.0
C2	3	74	7.4	3	1	8.4
C3	18	95	9.5	2	0	9.5
C4	31	125	12.5	2	0	12.5

Conclusion

We have implemented the GFV approach into our QA4J (Quality Analyst for Java) and applied the concepts in a couple of real projects. The experience has demonstrated that GFV addresses some of the problems initially identified related to effective usage of code review and human experts in quality assessments. In large engagements when customers have the some competency in articulating internal code quality expectations we have observed that this approach allows to capture those expectations precisely and formally and then continuously provide visibility into the structural fitness posture throughout the software construction and integration phase.

Further, we also deployed the approach to provide objective guidance to non-technical stakeholders in a software project related to the need for refactoring iterations to lower the *design debt (*to quote the term defined by Ward Cunningham related the design quality debt build up during software construction and maintenance) during a software project life-cycle.

The Software quality assurance phase within the life-cycle thus benefited from this approach since it complemented automated testing process that verifies functional fitness to requirements and QA4J now ensures structural fitness to internal code quality.

References

[BasiliWs1984] Victor Basili and D.M. Weiss: A Methodology for Collecting Valid Software Engineering Data, IEEE Transactions on Software Engineering, Vol. SE-10, no. 6, November 1984.

[Basili1992] Victor R. Basili, Software modeling and measurement: the Goal/Question/Metric paradigm, University of Maryland at College Park, College Park, MD, 1992

[Bloch2001] J. Bloch, Effective Java, Addison-Wesley, Reading, Massachusetts, 2001

[BMB1999] Lionel C.Briand, Sandro Morasca, Member, IEEE Computer Society, Victor R. Basili, Fellow, IEEE: Defining and Validating Measures for Object-Based High-Level Design, September 1999.

[Brooks1987] F. Brooks: No Silver Bullet: Essence and Accidents of Software Engineering, Computer Magazine, 1987

[ChK1994] Chindamber Shyam R., Kemerer Chris F: A Metric Suite for Object Oriented Design, IEEE Transactions on Software Engineering, June 1994

[ColemanEtal1994] - D.M. Coleman, D. Ash, B. Lowther, and P.W. Oman.: Using metrics to evaluate software system maintainability. IEEE Computer, 27(8):44--49, August 1994

[Fenton1991] Fenton N. E, Software Metrics A Rigorous Approach, Chapman and Hall.

[JohnMudit2006] John Kuriakose, Mudit Kaushik., Goal-Flavour Analysis – Predictable Structural Fitness for Code, submitted to SOQUA 2006, Third International Workshop on Software Quality Assurance, IEEE FSE 2006,

[Marinescu2002] Marinescu Radu, Measurement and Quality in Object-Oriented Design, Phd thesis, Faculty of Automatics and Computer Science of the Politehnica University of Timisoara, October 2002

[MensLz2002] T. Mens and M. Lanza,: A Graph-Based Metamodel for Object Oriented Software Metrics, Electronic Notes in Theoretical Computer Science, vol. 72, no. 2, 2002

[MeyerNiere] Matthias Meyer ,Software Engineering Group , University of Paderborn Germany and Jorg Niere Software Engineering Group , university of Siegen Germany Calculation and Visualization of Software Product Metrics

[SimonEtal2001] Frank Simon, Frank Steinbruckner, and Claus Lewerentz.: Metrics based refactoring. In Proc. European Conf. Software Maintenance and Reengineering, pages 30--38. IEEE Computer Society Press, 2001

[ZuseBl1989] Zuse, H. and P. Bollman: Software metrics: Using measurement theory to describe the properties and scales for software complexity metrics. SIGPLAN Notices 24 (8), 22—33, 1989

SOA-capability of software measurement tools

Martin Kunz, Andreas Schmietendorf, Reiner R. Dumke, Dmytro Rud

Software Engineering Group
Institute for Distributed Systems
University of Magdeburg, Germany
{makunz, schmiete, dumke, rud}@ivs.cs.uni-magdeburg.de
http://www.smlab.de

Abstract. In many areas of application in IT industry service-oriented architectures (SOA) [3] have proved their ability to create flexible structures with new functionalities by the integration of existing service offerings. In this paper the authors want to analyze the current situation in the area of software measurement tools with respect to which preconditions have to be fulfilled by measurement tools to easily integrate into a service-oriented measurement infrastructure.

Keywords: Service-oriented architectures, software measurement, SOA-capability, Service-oriented measurement infrastructure

1 Introduction

The significance of software measurement throughout the software development process is generally accepted, these days. Unfortunately, in practice common software measurement tools find small acceptance due to their high costs, inflexible structures, missing integration capabilities, and therewith unclear cost/benefit ratio.

In many different areas of application, service-oriented architecture has become the silver bullet to cover integration needs.

The requirements for measurement tools were derived out of a model for an ISO/IEC 15939 [2] based service-oriented measurement infrastructure which already was presented in [1] [5].

The analysis of existing measurement tools was performed with two different viewpoints:

a) A survey of manufacturers about existing products and existing integration opportunities

b) An independent analysis of existing measurement tools concerning the SOA-capability

Our detailed analysis covers aspects concerning among others functionality, interfaces, security, payment options, and supported standards.

As an additional result we identified three major capability levels of service interfaces:

1) Output of measurement values
2) Input of measurement data
3) Control and triggering of measurement services

In this way we consider two different views for interface evaluation: the used technology and the provided functionality.

In chapter 2 we want to present our assessment scheme for the independent analysis and the assessment results for over 30 different measurement tools.

In chapter 3 we describe the procedure for the survey of measurement tool manufactures and what the results can show us.

Subsuming, we point out needed steps to enhance measurement tools to create a service-oriented measurement infrastructure.

2 Assessment about SOA-capability of measurement tools

The target of this chapter is to analyze how far the aspects of Service-Oriented-Architectures have penetrated into the area of software measurement tools.

The SOA-capability depends predominantly on import potentiality of measurement data, export potentiality of results, and the implemented facilities to control the tool.

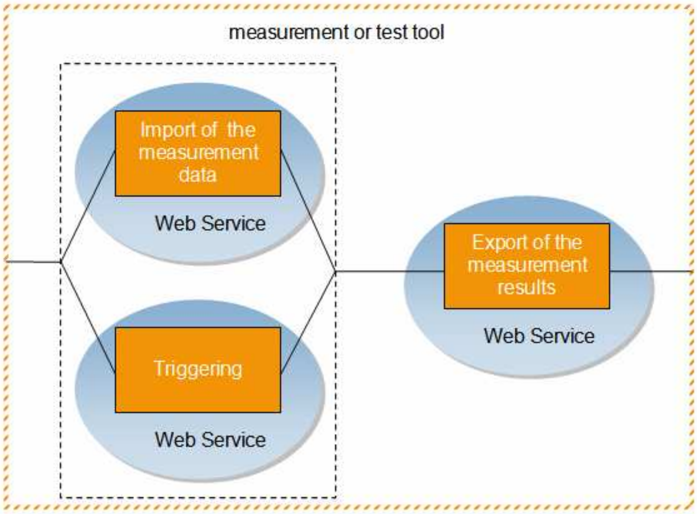


Figure 1: substantial functionality for SOA-capability

Since the most important thing for the usage of a distinct tool is the occurrence of interfaces for the export of measurement results, we focused our analysis on this aspect.

At first, we want to introduce a classification of interfaces according to the needed effort to adapt the interfaces into a service-oriented measurement infrastructure. We identify five major categories which are pointed out in the following table.

Existing Web-Service-Interface	*Very marginal effort*
XML-based Interfaces	*Low effort* (simple creation of Web-Services out of XML-files and databases)
Standardized exchange format	*Effort at medium level*
Proprietary exchange format	*Medium up to high effort*
No exchange possibility (only paper and printout)	*Continuous effort due to manual information transfer*

Table 1: Effort classification for SOA-execution of measurement results

By applying this classification approach to the 37 tools we have analyzed, one can see that over half of all tools just have a proprietary interface with a medium up to high adoption effort. The results are pointed out in figure 2 while the detail results for each tool are put into the appendix.

Based on the results one can reason that the majority of existing measurement tools have a medium or high effort for embedding them into an SOA. Only one third (34%) need just marginal or low effort.

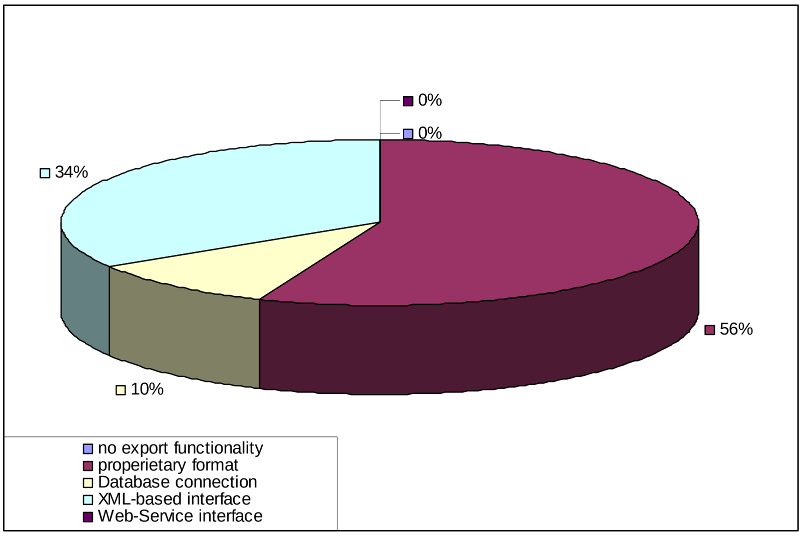


Figure 2: occurrence of different interface technologies

The second focus in the survey was on the combination of the functionality of different tools and/or the possibility to control the tool by an interface. This is important for the SOA idea of measurement tools because the main goal is to create a tailored measurement tool out of the functionality from different independent tools [1].

The results in this area are very uneven. Over half of all the tools just provide proprietary interfaces and in this way no possibilities exist to access functionality or to control the tool via the interface. Another 10% use database connections to transfer data in the first instance. These connections also do not have the precondition either to control or to access tool functionality.

Even if the other 34% do not provide options for controlling or accessing functionalities, the usage of XML at least allows enhancements in this direction in the near future.

As a subsumption one can say that the functionality of the presented interfaces does not deliver the needed requirements for service oriented architectures. In this way the realization of a software measurement tool as a Web-Service [7] offer is the exception.

Therefore, our second step was to make a survey among measurement tool manufactures as to how far their products are applicable to create Service-oriented measurement infrastructures [6], and if they are available as a service offer, respectively.

Another aspect was to find out if there are any planed ideas for SOA´s in future releases, if at the moment there is no service-oriented functionality.

3 Survey among measurement tool manufactures

As an outcome of our survey, we received over 30 replies. Some of the tools were part of our first assessment and, in general, one can outline that the answers give a representative view on software measurement tools from the manufactures point of view.

The first look is again on export interfaces. The results are shown in figure 3 and they are corresponding to the first assessment. Every tool has an export possibility but the effort to integrate them into SOA´s is not marginal due to non service-able connection type.

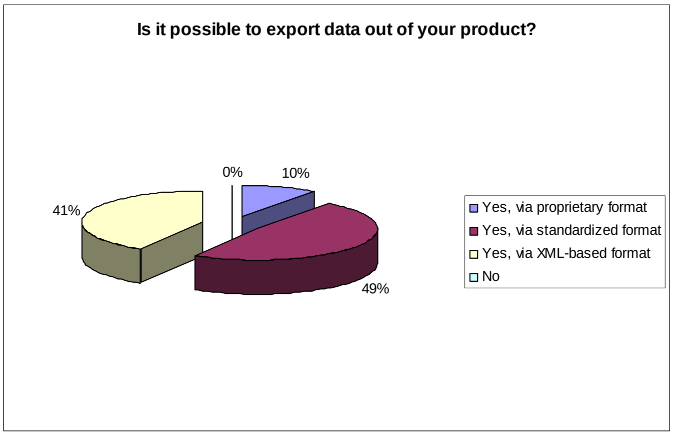


Figure 3: Results about export interfaces

The next questions target another important fact for Service-oriented Measurement Architectures: the combination of different tools according to distinct functionality. The results can be that for every distinct information need, different functionalities from different tools were combined to a new service [1] [5].

The results point out an optimistic view to this issue: over 80% of all manufactures say that their tool can be combined with other tools and nearly 80% say that single functionality of their tools can be used independently.

As we pointed out we can not approve this appraisal. The first reason is that tool manufactures often understand combining opportunities as the option to combine tools from their product portfolio and not to other tools e.g. by using open or standardized interfaces or connections. Because of the fact that we implicate more features into the concept of tool combination, we see our assessment not as rebut.

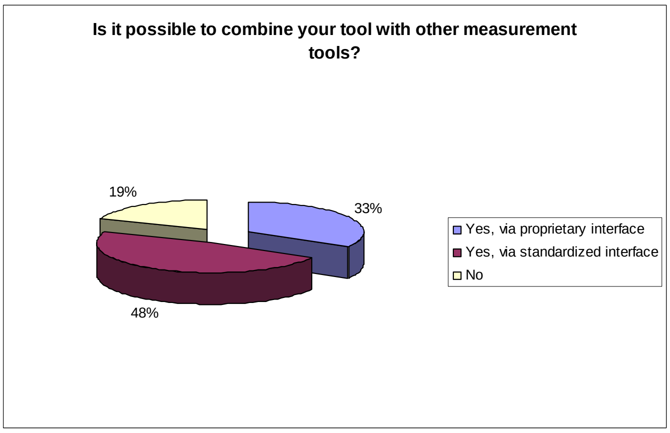


Figure 4: Combination of different measurement tools

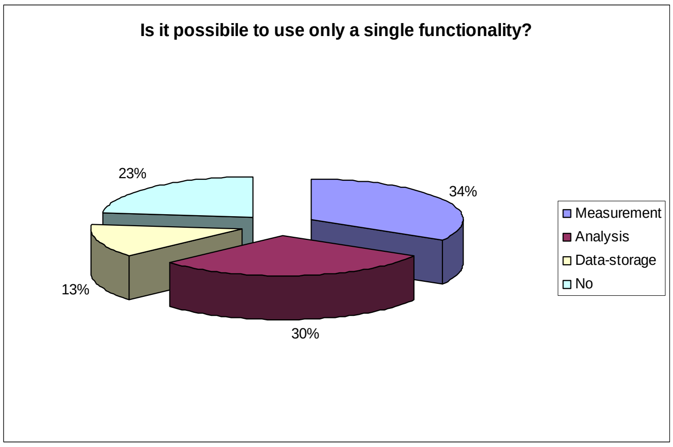


Figure 5: Usage of single functionality

Especially for commercial software, the question about the different type of license models is very important. If a software measurement tool provides functionality as a Web-service, the license model should provide pay-per-use to avoid high acquisition cost and therewith open up new possible users.

Unfortunately this aspect is not considered by the tool manufactures at the moment.

Only 3% provide a pay-per-use option and just a quarter provide at least the possibility to purchase a single module/component.

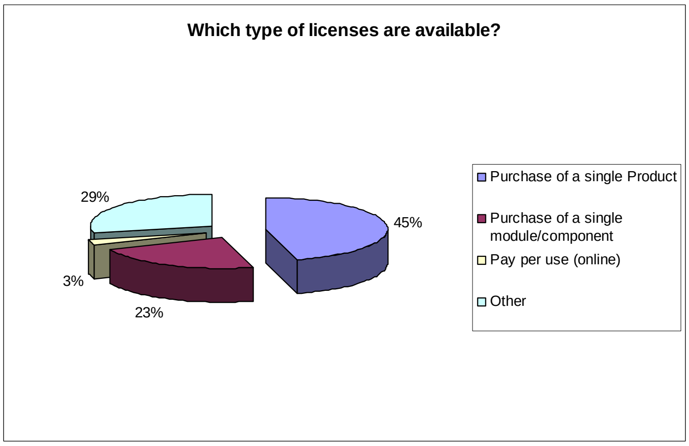


Figure 6: Different types of licenses models

4 Summary and Next Steps

Subsuming the assessment and the survey one can say that existing measurement tools are in general SOA-capable even if there is some effort to invest on.

But the establishment of new license models for service oriented tools is necessary in view of the current situation.

But to create a real benefit of service oriented architectures, the possibility of combination among different measurement tools has to be improved. Therefore, an accepted SOA Guideline which describes requirement and interface requirements has to establish and to create an applicable approach for service standardization and collaboration.

In general the technology of service oriented measurement architectures is not the silver bullet for the tool manufactures, at the moment. So it is up to the customers and users to ask for tailored functionality and new license models in the area of software measurement tools.

After research over measurement tools with an independent assessment and with a survey among tool manufactures, we should not forget the most important view on software measurement tools: the user perspective. A survey among measurement tool users is currently in progress and the results will give us new insights if service oriented measurement infrastructures are seminal.

Subsequently a preparation of measurement service guideline [8] is in process, containing information and proposals for service developers to create SOA-capable services.

References

1. Kunz, M., Schmietendorf, A., Dumke, R.R., Wille, C.: “Towards a service-oriented measurement infrastructure”, Proc. of the 3rd Software Measurement European Forum (SMEF), May 10-12, 2006, Rome, Italy, pp. 197-207
2. ISO/IEC, “ISO 15939 --- Information Technology --- Software Engineering Software Measurement Process” 2001
3. Erl, T. “Service-Oriented Architecture Concepts, Technology, and Design” Prentice Hall, 2005
4. Dumke, R.; Kunz, M.; Hegewald, H. & Yazbek, H. “An Agent-based Measurement Infrastructure” Shaker Publ. Proc. of the IWSM 2005, Montreal 2005
5. Peltz, C. “Web Services Orchestration and Choreography”, Computer, 2003
6. Schmietendorf, A., Dimitrov, E. “Implementation of services-oriented infrastructures by the use of a Service Centre” Proc. of the 2nd International Workshop on eServices and eLearning, Plovdiv 2004
7. Newcomer, E., Lomow, G. “Understanding SOA with Web Services” Addision Wesley, 2003
8. Schmietendorf, A.,Rud,D. “Performanceaspekte choreographierter Anwendungen auf Basis lose gekoppelter Web Services”, Proc. of the 6th Workshop „Performance Engineering in der Softwareentwicklung“ (PE 2005), 2005

Appendix:

Assessment results:

Measurement tool	**Non exportable report**	**Non exportable charts**	**Results presented in HTML**	**File** (Format)	**DB (Interface (JDBC, …))**	**XML**	**Web-Service-oriented**
JMetrics	x	x	*(planned: XML to HTML Reports)	x (Textfile)	-	*(planned)	-
Imagix 4D	x	x	x	x - RTF, -CSV [Microsoft Visio-Support]) - PS - PNG -Table (ASCII text file)	-	-	-
SLIM-Metrics and SLIM-DataManager	x	x	-	-	x (DB, ODBC)	-	-
Scientific Toolworks, Inc. **Understand for ADA** **Understand for C++** **Understand for Delphie** **Understand for FORTRAN** **Understand for Java** **Understand for JOVIAL**	x	x	x	x (PERL API, C/C++ API)	-	-	-
Function Point Workbench	x	x	x	x (spreadsheet format)	-	x (XML-File)	-
TAU / Logiscope	x	x	x	x (Word, Framemaker, Interleaf)		??	
SCOPE - Project Sizing Software	x	x	-	x (Word, Excel, PDF)	x (MsACCESS)	-	-
SDMetrics	x	x	x	x (text [tab seperated tables], Openoffice.org CALC, XML for Microsoft	-	x (XML for Microsoft Excel)	-

				Excel)			
CMM-Quest	x	x	x	x (text)	-	-	-
CMTJava, CMT++	x	x	x	x (text, Excel, XML)	-	x	-
COSTAR	x	x	-	x (text (Costar's native file format), CSV, Excel, BMP)	-	-	-
EPM – Essential Project Manager	x	x	x	x (CSV)	* (SQL DB, JDBC)	x	-
EM – Essential Metrics	x	x	x	x (CSV)	* (SQL DB, JDBC)	x	-
Krakatau Essential Metrics	x	x	x	x (CSV)	x (MySQL data file)	x	-
Krakatau Project Manager	x	x	x	x (CDF/CSV)	-	-	-
Krakatau Professional **Krakatau Lite**	x	x	x	x (CDF/CSV)	-	-	-
Ressource Standard Metrics – RSM / RSM Wizard (GUI)	x	-	x	x (text, CSV)	-	-	-
TychoMetrics	x	x	x	x (CSV, MetaFile, BMP, JPG, PNG)	- (x) [not confirmed]	- (x) [not confirmed]	-
Appraisal Wizard	x	x	x	x (text, CSV, Excel, rtf, pdf, bmp, jpg, wmf, emf, gif, wb1, wk2, dif, slk)	-	-	-
Metrics4C, Metrics4FORTRAN, Metrics4Pascal, Metrics4Project	x	x	-	x (text[console output], gif)	-	-	-
ASSESS für Assembler, COBOL, Java	x	-	x	x (with external Plugins: CSV,…)	x (MSAccess)	x	-
ExperiencePro	x	x	x	x (txt, xls)	-	-	-
CodeReports	x	x	x	x (xls, Crystal Reports, …)	x (ODBC, JDBC)	-	-
CodeCheck	x	-	*(customizeable)	x (txt) *customizeable	-	* (customizeable)	-
MPP (Kuhrau)	x	-	-	x (txt,xls)	-	-	-

COSMOS	x	-	-	x (txt)	-	-	-
LDRA Testbed	x	x	x	x (txt)	-	-	-
JUNIT	x	x	x	x (txt)	-	x	-
JDepend	x	x	x (with Ant)	x (txt)	-	x	-

Improving Logistic Regression Predictions of Software Quality using Principal Component Analysis

K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra

University School of Information Technology, GGS Indraprastha University, Delhi 110006, India

{kka@ipu.edu, yy66@rediffmail.com, arvinderkaurtakkar@yahoo.com, ruchikamalhotra2004@yahoo.com}

Abstract. Assessment of object-oriented systems requires robust modeling techniques. Multivariate models build to predict fault proneness can be misleading if the metrics are highly correlated Khoshgaftaar et al. propose using principal component analysis to avoid such problems. The advantages of using this technique have not been empirically demonstrated especially for object-oriented systems. Our study illustrates the use of principal component analysis to predict software quality models. We empirically explore relationship between object-oriented metrics given by Chidamber and Kamerer and Fault proneness. This study used data collected from Java applications. The study used logistic regression analysis for predicting fault prone classes. The results show that principal component analysis can improve predictive quality of software model.

Keywords: Software quality, Measurement, Metrics, Object-Oriented, Coupling, Cohesion, Inheritance, Principal component analysis

1 Introduction

Assessment of Object-Oriented (OO) systems is an important element in the process of assuring the high quality software. There are several metrics proposed in the literature to capture the quality of OO design and code, for example, (Aggarwal et al.[4]; Briand et al., [10, 11]; Bieman and Kang [8]; Cartwright and Shepperd [13]; Chidamber and Kamerer[14, 15]; Harrison et al. [20]; Henderson-sellers [21]; Hitz and Montazeri [22]; Lake and Cook [26]; Li and Henry [28]; Lee et al. [27] Lorenz and Kidd [29]; Tegarden et al [335]). These metrics provide ways to evaluate the quality of software and their use in earlier phases of software development can help the organizations in assessing a large software development quickly, at a low cost. The application of statistical modeling techniques has been a widely pursued area of research in predicting software quality models. The intense pressure of competition among organizations and the cost of software maintenance are often cited as primary motivators of improving the method of prediction of software quality.

Any improvement in the quality of the software developed and software development process depends on quality predicting to the highest degree of accuracy possible. Logistic Regression (LR) modeling is being widely applied to the field of software engineering. The results reported that LR models have high accuracy in predicting the faulty classes. (Basili et al. [5]; Binkley and Schach [9]; Briand et al [12]; Chidamber and Kamerer [16]; El Emam et al. [17]; Gyimothy et al. [18]; Ping et al. [31]).

All the modeling techniques have limitations. Multivariate models used to predict quality attributes can be misleading if the OO metrics are highly correlated. Mumsum and Khoshgaftaar [30] proposed using principal component analysis to transform OO metrics into orthogonal variables. Although there is no guarantee of improved results, but in practice one often finds that raw metrics are highly correlated to each other. Although other studies have illustrated the use of application of principal component analysis, we are unaware of application of principal component analysis in OO systems. This paper uses LR for predicting fault prone classes. We empirically validate metrics given by Chidamber and Kamerer [14] on systems developed using Java language. In our study we compare two models: one based on set of OO metrics and other based on principal component analysis.

The paper is organized as follows: Section 2 summarizes the metrics studied and describes sources from which data is collected. Section 3 presents the research methodology followed in this paper. The results of the study are given in section 4. The model is evaluated in section 5. Conclusions of the research are presented in section 6.

2 Research Background

In this section we present the summary of metrics studied in this paper (Section 2.1) and empirical data collection (Section 2.2).

2.1 Dependent and Independent Variables

The binary dependent variable in our study is fault proneness. The goal of our study is to empirically explore the relationship between OO metrics and fault proneness at the class level. Fault proneness is defined as the probability of fault detection in a class. We use logistic regression to predict probability of fault proneness. Our dependent variable will be predicted based on the faults found in testing phase during software development. The metrics given by Chidamber and Kamerer [14] are summarized in Table 1.

Table 1. Metrics Studied [14, 2, 3]

Metric	Definition
Coupling between Objects (CBO)	CBO for a class is count of the number of other classes to which it is coupled and vice versa.
Lack of Cohesion (LCOM)	It counts number of null pairs of methods that do not have common attributes.
Number of Children (NOC)	The NOC is the number of immediate subclasses of a class in a hierarchy.
Depth of Inheritance (DIT)	The depth of a class within the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree and is measured by the number of ancestor classes.
Weighted Methods per Class (WMC)	The WMC is a count of sum of complexities of all methods in a class. Consider a class K1, with methods M1,…….. Mn that are defined in the class. Let C1,……….Cn be the complexity of the methods. $\mathrm{WMC} = \sum_{i=1}^{n} C_i$
Response for a Class (RFC)	The response set of a class (RFC) is defined as set of methods that can be potentially executed in response to a message received by an object of that class. It is given by RFC=|RS|, where RS, the response set of the class, is given by $\mathrm{RS} = M_i \cup_{all\ j} \{R_{ij}\}$
Lines Of Code (LOC)	It counts the lines of code.

2.2 Empirical Data Collection

To analyze metrics chosen for this work, their values are computed for twelve different systems consisting of 136 system classes (39KLOC). These systems are developed by under graduate engineering students and Masters of Computer Application students at School of Information Technology, Guru Gobind Singh Indraprastha University. The systems are developed using Java programming language. All students had experience with Java language and thus they had basic knowledge necessary for this study.

The students were divided into 12 teams of four students each. Each team developed a medium-sized system such as flight reservation, chat server, proxy server etc. The development process uses waterfall model. Documents were produced at each phase of software development. A separate group of students having prior knowledge of system testing under the guidance of senior lecturers are assigned the task of testing systems according to test plans and black-box testing techniques. The test plans were made using Rational software.

The following data is collected:

1. The design and source code of the java programs
2. The faulty data found by the testing team.

3 Research Methodology

In this section the steps taken to analyze coupling, cohesion, inheritance and size metrics for classes taken for analysis are described. The procedure used to analyze the data collected for each measure is described in following stages (i) descriptive statistics and outlier analysis (ii) Principal Component (P.C.) analysis (iii) Logistic Regression (LR) modeling (iv) model evaluation.

I. Descriptive Statistics and Outlier analysis: The research data must be suitably reduced so that the same can be read easily and can be used for further analysis. Descriptive statistics concern development of certain indices or measures to summarize data. The important statistics measures used for comparing different case studies include mean, median, and standard deviation. Low variance metrics do not differentiate classes very well and therefore are not likely to be useful, and hence are excluded from our study. Data points, which are located in an empty part of the sample space, are called outliers. Univariate and multivariate outliers are found in our study. To identify multivariate outlier we calculate for each data point the Mahalanobis Jackknife distance. Outlier analysis is done to find data points that are over influential and removing them is essential. Details on outlier analysis can be found in [6].

II. Principal-Component (or P.C.) Analysis: Many OO metrics have high correlation with each other. P.C analysis transforms raw metrics to variables that are not correlated to each other when the original data are OO metrics, we call the new P.C. variables domain metrics [25].

P.C. analysis is used to maximize the sum of squared loadings of each factor extracted in turn [24]. The P.C. analysis aims at constructing new variable (P_i), called Principal Component (P.C.) out of a given set of variables $X_j's(j = 1,2,....,k)$

$$\begin{aligned} P_1 &= b_{11}X_1 + b_{12}X_2 + ... + b_{1k}X_k \\ P_2 &= b_{21}X_1 + b_{22}X_2 + ... + b_{2k}X_k \\ &. \quad . \quad . \quad . \quad . \\ P_k &= b_{k1}X_1 + b_{k2}X_2 + ... + b_{kk}X_k. \end{aligned} \tag{1}$$

All b_{ij}'s called loadings are worked out in such a way that the extracted P.C.s satisfy the following two conditions:

(i) P.C.s are uncorrelated (orthogonal) and

(ii) The first P.C. (P_1) has the highest variance, the second P.C. has the next highest variance so on.

The variables with high loadings help identify the dimension P.C. is capturing, but this usually requires some degree of interpretation. In order to identify these variables, and interpret the P.C.s, we consider the rotated components. As the dimensions are independent, orthogonal rotation is used. There are various strategies to perform such rotation. We used the varimax rotation, which is the most frequently used strategy in literature. Eigenvalue or latent root is associated with P.C., when we take the sum of squared values of loadings relating to dimension, then the sum is referred to as eigenvalue. Eigenvalue indicates the relative importance of each dimension for the particular set of variables being analyzed. The P.C.s with eigenvalue greater than 1 are taken for interpretation.

III. Model prediction using Logistic Regression (LR): LR is the most widely used technique [23] in literature is used to predict dependent variable from set of independent variables (a detailed description is given by [23] and [5]. Binary LR is used to construct models when the dependent variable is binary as in our case. In our study, the dependent variable is fault proneness and the independent variable is OO metrics. LR is of two types: (i) Univariate LR (ii) Multivariate LR

Univariate LR is a statistical method that formulates a mathematical model depicting relationship among dependent variable and each independent variable. Multivariate LR is used to construct a prediction model for the fault-proneness of classes. In this method metrics are used in combination. The multivariate LR formula can be defined as follows:

$$prob(X_1, X_2, ...X_n) = \frac{e^{(A_o + A_1X_1 + ...+ A_nX_n)}}{1 + e^{(A_o + A_1X_1 + ...+ A_nX_n)}} \quad (2)$$

where $X_i, i = 1,2,.......,n$, are the independent variables. *Prob* is the probability of detecting faults in a class. In LR two stepwise selection methods forward selection and backward elimination can be used [23]. Stepwise variable entry examines the variables in the block at each step for entry. This is a forward stepwise procedure. The backward elimination method includes all the independent variables in the model. Variables are deleted one at a time from the model until a stopping criteria is fulfilled. We have used backward elimination method using metrics selected in P.C. analysis and univariate analysis.

For model predicted we performed a test of multicollinearity. The interpretation of model becomes difficult if multicollinearity is present. Let $X_1, X_2, ...X_n$ be the covariates of the model predicted. P.C. analysis is applied on these variables to find maximum eigenvalue, e_{max} and minimum eigenvalue, e_{min}. The conditional number is defined as $\lambda = \sqrt{e_{\mathbf{max}} / e_{\mathbf{min}}}$. If the value of the conditional number is 30 then multicollinearity is not tolerable [7].

The following statistics are reported for each significant metric:

- **Maximum Likelihood Estimation (MLE) and Coefficients (A_i's):** MLE is a statistical method for estimating the coefficients of a model. The likelihood function (*L*) measures the probability of observing the set of dependent variable values (P_1, P_2… P_n). It is written as the probability of the product of the independent variables:

$$L = prob(P_1 * P_2 * ...... * P_n) \qquad (3)$$

MLE involves finding the coefficients that makes the log of the likelihood function $Log(L)$ as large as possible. The larger the value of coefficients larger is the impact of the independent variables (OO metrics) on the predicted fault-proneness.

- **The statistical significance (S*ig.*):** It measures the significance level of the coefficient, larger the statistical significance less is the estimated impact of the independent variables (OO metrics). In our study we used 0.05 as the significance threshold. We made this choice because it is an indirect means to control the number of variables in the final model.
- **The R^2 Statistic:** It is the proportion of the variance in the dependent variable that is explained by the variance of the independent variables. The higher the effect of the model's independent variables more is the accuracy of the model.

IV. Evaluating the performance of the Model: The criterion to assess the quality of a predicted model in our study is discrimination. Discrimination refers to the ability of a model to distinguish those classes that are faulty from those that are non faulty. A perfectly discriminating model would assign a higher probability to faulty classes than to that are non faulty classes. Common measures of discrimination are:

- The sensitivity and specificity of the model is calculated to predict the correctness of the model. The percentage of classes correctly predicted to be fault prone is known as sensitivity of the model. *Sensitivity* can be formally defined as:

$$\text{S}enstivity\ (s) = \frac{\text{Classes correctly predicted as fault prone}}{\text{Classes actually fault prone}} \qquad (5)$$

The bigger the sensitivity, the better the model. The percentage of non-occurrences correctly predicted i.e. classes predicted not to be fault prone is called specificity of the model. *Specificity* can be formally defined as:

$$\text{S}\mathit{pecificity}\ (f) = \frac{\text{Classes correctly predicted not fault prone}}{\text{Classes actually not fault prone}} \quad (6)$$

Ideally both the sensitivity and specificity should be high. A low sensitivity means that there are many low risk classes that are classified as faulty. Therefore, the organization would waste resource in focusing additional testing effort on these classes. A low specificity means that there are many high risk classes that are classified as not faulty. Therefore, the organization would be passing high risk classes to customers [17]. We report sensitivity and specificity of the predicted models for a single threshold 0.5.

- Yourdon's *J* coefficient [17]: *J* coefficient is defined as:

$$J = s + f - 1 \quad (7)$$

The J coefficient can vary from –1 to +1 with plus 1 being perfect accuracy and –1 being the worst accuracy.

- True Positive Rate (TPR) [17]: TPR is defined as:

$$\begin{aligned}&\text{Let n1} = \text{Number of classes falsely predicted as fault prone}\\&\text{a1} = \text{Number of classes correctly classified as fault prone}\\&\text{a} = \text{Total number of classes}\end{aligned}$$

$$\text{TPR} = \frac{\text{n1} + \text{a1}}{\text{a}} \quad (8)$$

- Proportion correct is defined as:

$$\begin{aligned}&\text{Let a1} = \text{Number of classes correctly classified as fault prone}\\&\text{a2} = \text{Number of classes correctly classified as not fault prone}\\&\text{a} = \text{Total number of classes}\end{aligned}$$

$$\text{Proportion Correct} = \frac{a1 + \text{a2}}{\text{a}} \quad (9)$$

- Receiver Operating Characteristic (ROC) analysis: The LR model outputs were evaluated for performance using ROC analysis. ROC curve, which is defined as a plot of sensitivity as the y-coordinate versus its 1-specificity as the x coordinate, is an effective method of evaluating the quality or performance of predicted models [17]. While constructing ROC curves, one selects many cutoff points between 0 and 1 in our case, and calculates sensitivity and specificity at each cut off point.

Area under the ROC curve (AUC) is a combined measure of sensitivity and specificity [19]. To compute the accuracy of the predicted models, we use the area under the ROC curve. The standard error for ROC curves was determined according to the method proposed by Hanley and McNeil [19]:

$$SE = \sqrt{\frac{AUC(1-AUC)+(n_m-1)(Q_1-AUC^2)+(n_b-1)(Q_2-AUC^2)}{n_m n_b}} \qquad (10)$$

$$Q_1 - \frac{AUC}{2-AUC}, Q_2 = \frac{2AUC^2}{1+AUC}$$

- To predict the accuracy of model it should be applied on different data sets. We therefore performed k-cross validation of model [32]. The data set is randomly divided into k subsets. Each time one of the k subsets is used as the test set and the other k-1 subsets are used to form a training set. Therefore, we get the fault proneness for all the k classes.

4 Analyses Results

In this section we will describe the analyses we performed to find the relationship between OO metrics and fault proneness of the classes. We first employed LR [23] to study the relationship between OO metrics and fault proneness of classes. We then employed LR to study the relationship between domain (found by using P.C. analysis) metrics and fault proneness of classes. This method is rarely applied in this area.

In these methods we applied multivariate to determine combined effect of OO metrics on fault proneness.

4.1 Descriptive Statistics

Each table presented in the following subsections show "min", "max", "mean", "std dev", "variance", "25% quartile" and "75% quartile" for all metrics considered in this study.

Table 2. Descriptive Statistics for Size metrics

Metric	Min.	Max.	Mean	Std. Dev.	Variance	Percentile (25%)	Percentile (75%)
CBO	0	8	1.058	1.312	1.722	0	1
DIT	0	3	0.352	0.611	0.373	0	1
NOC	0	6	0.282	0.958	0.919	0	0
LOC	4	707	114.42	151.65	23000	28.5	116.5
RFC	0	85	85	13.4	16.357	3	20
WMC	0	31	31	5.964	6.889	1	8
LCOM	0	336	25.56	65.66	4311.5	0	13.5

The following observations are made from Table 2:

- The size of a class measured in terms of lines of source code ranges from 4-707.
- The values of DIT and NOC are less, which shows that the inheritance is not much used in all the systems; similar results have also been shown by other studies (Chidamber and Kamerer [14] Briand et al [12], Cartwright and Shepperd [13]).
- The LCOM measure, which counts the number of classes with no attribute usage in common, has very high values (upto 336).

4.2 Principal Component (P.C.) Analysis

In this section the results of applying P.C. analysis are presented. The P.C. extraction analysis and varimax rotation method is applied on all metrics. The rotated component matrix is given in Table 3. The values above 0.7 (shown in bold in Table 3) are the metrics that are used to interpret the PCs. For each PC, we also provide its eigenvalue, variance percent and cumulative percent. The interpretations of PCs are given as follows:

- P1: RFC, LCOM, WMC, and LOC are cohesion, coupling and size metrics. We have size, coupling and cohesion metrics in this dimension. This shows that there are classes with high internal methods (methods defined in the class) and external methods (methods called by the class). This means cohesion and coupling is related to number of methods and attributes in the class.
- P2: CBO, DIT are coupling and inheritance metrics.

Table 3. Rotated Principal Components

P.C.	**P1**	**P2**
CBO	0.2242	**-0.6633**
DIT	-0.043	**0.6963**
LCOM	**0.8100**	-0.072
LOC	**0.8685**	0.0087
NOC	0.2911	0.4451
RFC	**0.8608**	0.0168
WMC	**0.9217**	-0.0664

4.3 Logistic Regression (LR) Analysis

In this section we present the results of multivariate analysis using raw metrics and domain metrics. The conditional number for the models is which is well below 30, which implies that multicollinearity of predicted models is very much tolerable. None of the outlier was found to be influential during univariate and multivariate analysis

Model I: Model with all metrics

Table 4 predicts accuracy of the model containing OO metrics, using backward elimination method. Three metrics RFC, WMC, and CBO are included in the model. All of the metrics were found to be significant in univariate analysis.

Table 4. Multivariate Analysis for Model I

Variable	**RFC**	**WMC**	**CBO**	**Constant**
B	0.1109	0.1682	0.7401	-3.1410
p-value	0.0074	0.0326	0.048	0.000
-2 Log likelihood: 64				
R^2 Statistic: 0.44				

As shown in Table 5, a threshold of P_0=0.5 is chosen. Classes with predicted probability above 0.5 are classified to be fault prone and below this threshold are classified as not to be fault prone. This threshold was selected to balance the number of actual and predicted faults. Out of 37 classes actually fault prone, 26 classes were predicted to be fault prone. The sensitivity of the model is 70.27%. Similarly 41 out of 48 classes were predicted not to be fault prone. Thus specificity of the model is 85.42%. This shows that the model correctness is high.

Table 5. Predicted Correctness of Model I

		Predicted	
		Not Faulty	**Faulty**
		P_o<=0.5	Po>0.5
Observed	**Not Faulty**	41	7
	Faulty	11	26

Model II: Model with Principal Component Metrics

Table 6 predicts accuracy of the model containing domain metrics, using backward elimination method.

Table 6. Statistics for Model II

Variable	**P1**	**P2**	**Constant**
Coefficient	3.4527	-0.9325	0.3684
p-value	0.000	0.0226	0.3458
-2 Log likelihood: 63.4			
R^2 Statistic: 0.456			

Out of 37 classes actually fault prone, 31 classes were predicted to be fault prone. The sensitivity of the model is 75.68%. Similarly 46 out of 48 classes were predicted not to be fault prone. Thus specificity of the model is 87.50%. This shows that the Model II has better accuracy as compared to Model I.

Table 7. Predicted Correctness of Model II

		Predicted	
		Not Faulty	**Faulty**
		P_o<=0.5	Po>0.5
Observed	**Not Faulty**	42	6
	Faulty	9	28

5 MODEL EVALUATION

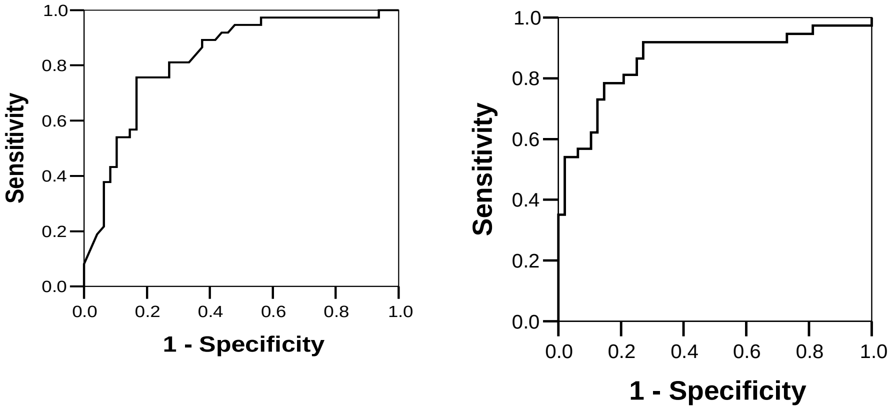
The accuracy of model predicted is quite high but it is somewhat optimistic since the model is applied on same data set from which it is derived from. To predict accuracy of model it should be applied on different data sets thus we performed 9-cross validation of Models following the procedure given in Section 3. For the 9-cross validation, the classes were randomly divided into 9 parts of approximately equal (5 partitions of 9 data points each and 4 partitions of 10 data points each). We summarized the results of cross validation of predicted models via logistic regression approach in Table 9.

Table 8. Results of 9-cross validation of Models

Technique	Sensitivity	Specificity	Proportion correct	J Coefficient	TPR	AUC
Model I	70.2	83.3	78.82	0.535	77.4	0.826
Model II	72.97	85.4	80	0.583	76.4	0.865

In Figure 1, the ROC curves for Model I and Model II are presented. The ROC curve for the Model I is shown in Figure 1(a), the area under the ROC curve, AUC, was 0.826 (SE 0.0472), providing 70.2% of sensitivity and 83.3% of specificity. Whereas, the area under the ROC curve for Model II was 0.865 (SE 0.0442), with sensitivity 72.97% and specificity 85.4%.

As shown in Table 8, the results of cross validation of Model II (with domain metrics) were better compared to cross validation results of Model I (with raw metrics) using LR analysis. According to the results of this study, the difference in ROC areas (AUC) between Model II and Model I is 3.9%.



(a) (b)

Fig. 1. ROC curve for (a) Model I and (b) Model II

6 Conclusions

This study illustrates that applying the principal component analysis can improve a software quality model, compared to a similar model based on raw metrics. We applied these models on 12 projects and employed Logistic regression methods to assess the applicability of the models. Given data on a past project, if a model similar to Model II was built during the design or coding phase of development, the model would predict which classes are likely to be fault prone (not fault prone) with better accuracy compared to Model I.

While research continues, the practitioner could apply the metrics captured in predicted model to predict faulty classes. We can assess the quality of the software in the earlier stages of software development by computing the values of metrics found in predicted model. The classes predicted to be fault prone will need extra attention during rest of the development. Planning and resource allocating for inspection and testing is difficult and it is usually done on empirical basis. The model predicted in the above section could be of great help for planning and executing the testing activities. One important application of predicted models is also to build quality benchmarks to assess OO constructs in OO systems that are newly developed or under maintenance e.g. in the case of software acquisition and outsourcing. Thus we can conclude that predicted model appear to be well suited to develop the practical quality benchmark.

As in all empirical studies the relationship we established is valid only for certain population of systems. In our case, we can roughly characterize this population as 'object-oriented, medium-sized systems'.

The metrics could not be evaluated over a large data set but this is a problem that has plagued much of empirical software engineering research. More similar type of studies must be carried out with different data sets to give generalized results across different organizations. We plan to replicate our study on large data set and industrial OO software system. We further plan to replicate our study to predict models based on the early analysis and design artifacts

References

[1] Aggarwal, K. K., Singh, Y., Kaur, A: Neural Net Based Approach to Test Oracle", ACM SIGSOFT, vol. 29, issue 3, (2004).

[2] Aggarwal, K. K., Singh, Y., Kaur, A and Malhotra, R.: Analysis of Object-Oriented Metrics. In Proceedings of the International Workshop on Software Measurement (IWSM), Montréal, Canada (2005).

[3] Aggarwal, K. K., Singh, Y., Kaur, A and Malhotra, R.: Empirical study of object-oriented metrics. Journal of Object Technology.

[4] Aggarwal, K. K., Singh, Y., Kaur, A and Malhotra, R.: Software reuse metrics for object-oriented systems. In Proceedings of the Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA '05)(2005) 48-55.

[5] Basili,V., Briand, L. and Melo, W.: A validation of object-oriented design metrics as quality indicators. IEEE Transactions on Software Engineering 22, no. 10, (1996) 751-761.

[6] Barnett , V., Price , T.: Outliers in Statistical Data, John Wiley & Sons (1995).

[7] Belsley, D., Kuh, E. and Welsch, R.: Regression diagnostics: Identifying influential data and sources of collinearity. New York: John Wiley and Sons (1980).

[8] Bieman, J., and Kang B. Cohesion and reuse in an object-oriented system. In Proceedings of the ACM Symposium on Software Reusability (SSR'94) (1994) 259-262.

[9] Binkley, A., and Schach, S.: Validation of the coupling dependency metric as a risk predictor. In Proceedings of the International Conference on Software Engineering (1998) 452-455.

[10] Briand, L., Daly, W. and Wust, J.: Unified framework for cohesion measurement in object-oriented systems. Empirical Software Engineering 3 (1998) 65-117.

[11] Briand, L., Daly, W. and Wust, J.: A unified framework for coupling measurement in object-oriented systems. IEEE Transactions on Software Engineering 25 (1999) 91-121.

[12] Briand, L., Daly, W. and Wust, J.: Exploring the relationships between design measures and software quality. Journal of Systems and Software 5 (2000) 245-273.

[13] Cartwright, M., and Shepperd, M.: An empirical investigation of an object-oriented software system. IEEE Transactions of Software Engineering 26, no.8 (1999) 786-796.

[14] Chidamber, S. R., and Kamerer, C. F.: A metrics suite for object-oriented design. IEEE Transactions on Software Engineering SE-20, no. 6 (1994) 476-493.

[15] Chidamber, S., and Kemerer, C.: Towards a metrics suite for object oriented design. In Proceedings of the Conference on Object-Oriented Programming: Systems, Languages and Applications (OOPSLA'91). SIGPLAN Notices 26 no. 11 (1991) 197-211.

[16] Chidamber, S., Darcy, D., and Kemerer, C.: Managerial use of metrics for object-oriented software: An exploratory analysis. IEEE Transactions on Software Engineering 24, no. 8 (1998) 629-639.

[17] Emam , K.El, Benlarbi , S., Goel N., Rai. :A Validation of Object-Oriented Metrics, Technical Report ERB-1063, NRC (1999).

[18] Gyimothy , T., Ferenc , R., Siket, I.: Empirical validation of object-oriented metrics on open source software for fault prediction", IEEE Trans. Software Engineering, vol. 31, Issue 10 (2005) 897 – 910,

[19] Hanley J, McNeil, BJ.: The meaning and use of the area under a Receiver Operating Characteristic ROC curve, Radiology, vol. 143 (1982). 29-36.

[20] Harrison, R., Counsell, S. J. and Nithi, R.V.: An evaluation of MOOD set of object-oriented software metrics. IEEE Transactions on Software Engineering SE-24, no. 6 (1998) 491-496.

[21] Henderson-Sellers, B.: Object-oriented metrics, measures of complexity. Englewood Cliffs, N.J.: Prentice Hall (1996).

[22] Hitz, M., and Montazeri.: Measuring coupling and cohesion in object-oriented systems. In Proceedings of the International Symposium on Applied Corporate Computing, Monterrey, Mexico (1995).

[23] Hosmer, D., and Lemeshow, S.: Applied logistic regression. New York: John Wiley and Sons (1989).

[24] Kothari, C. R.: Research Methodology. Methods and Techniques. New Delhi**:** New Age International Limited (2004).

[25] Khoshgaftaar, T.M., Allen, E.D., Hudepohl, J.P, Aud, S.J.: Application of neural networks to software quality modeling of a very large telecommunications system, IEEE Transactions on Neural Networks, Vol. 8, No. 4 (1997) 902--909.

[26] Lake, A., and Cook, C.: Use of factor analysis to develop OOP software complexity metrics. In Proceedings of the 6th Annual Oregon Workshop on Software Metrics, Silver Falls, Oregon (1994).

[27] Lee, Y., Liang, B., Wu, S. and Wang, F.: Measuring the coupling and cohesion of an object-oriented program based on information flow. In Proceedings of the International Conference on Software Quality, Maribor, Slovenia (1995).

[28] Li, W., and Henry, S.: Object-oriented metrics that predict maintainability. Journal of Systems and Software 23, no. 2 (1993) 111-122.

[29] Lorenz, M., and Kidd, J.: Object-oriented software metrics. Englewood Cliffs, N.J.: Prentice-Hall (1994).

[30] Munson, J.C. and. Khoshgaftaar, T.M.: The detection of fault prone programs, IEEE Transactions in Software Engineering, Vol. 18 (1992)423-433.

[31] Ping, Yu., Xiaoxing, Ma., Jian, Lu.: Predicting Fault-Proneness using OO Metrics: An Industrial Case Study, CSMR 2002, Budapest, Hungary (2002) 99-107.

[32] Stone, M.: Cross-validatory choice and assessment of statistical predictions. J. Royal Stat. Soc. 36: (1974) 111-147.

[33] Tegarden, D., Sheetz, S., and Monarchi, D.: A software complexity model of object-oriented systems. Decision Support Systems 13 (3-4) (1995) 241-262.

Web Services based Measurement for IT Quality Assurance

Ayaz Farooq[1], Steffen Kernchen[1], Reiner R. Dumke[1], Cornelius Wille[2]

[1] Institute for Distributed Systems, University of Magdeburg
P.O. Box 4120, D-39106 Magdeburg, Germany
(farooq,kernchen,dumke)@ivs.cs.uni-magdeburg.de
[2] High Business School, Bingen, Germany wille@fh-bingen.de

Abstract. Concern for quality assurance in information technology is growing with every passing day. Software measurement encompasses all aspects of IT including product, process and resources and provides us adequate quantitative means for assessing various artifacts involved in information technology. Service-oriented architecture provides flexibility, reliability and scalability and is gaining tremendous potential. Conventional hard-wired software measurement and analysis tools are expensive and limited in functionality and ease of use. This situation can be improved by software measurement web services which can provide us an easy and flexible approach towards measurement-based quality evaluation of software products. We present the idea of service-oriented approach for IT quality assurance and introduce a software measurement web service for evaluating and analyzing software products. We apply our approach, using our web service as a case study, to evaluate and analyze Java software products and libraries thereby highlighting their various quality aspects.

Keywords: Service-orientation, software quality assurance, software measurement, web services, object-oriented metrics, Java

1 Introduction

Software measurement is a key area in IT quality assurance. Software measurement process is an umbrella activity that covers all areas and aspects relating to software product, process and resources. The ISO/IEC 15939 Information Technology - Software Measurement Process standard [1] (see figure 1) helps us to identify, define, select, apply, and improve software measurement in any software development project. This standard describes a compliant measurement process concerning its purposes and outcomes combined with appropriate activities and tasks.

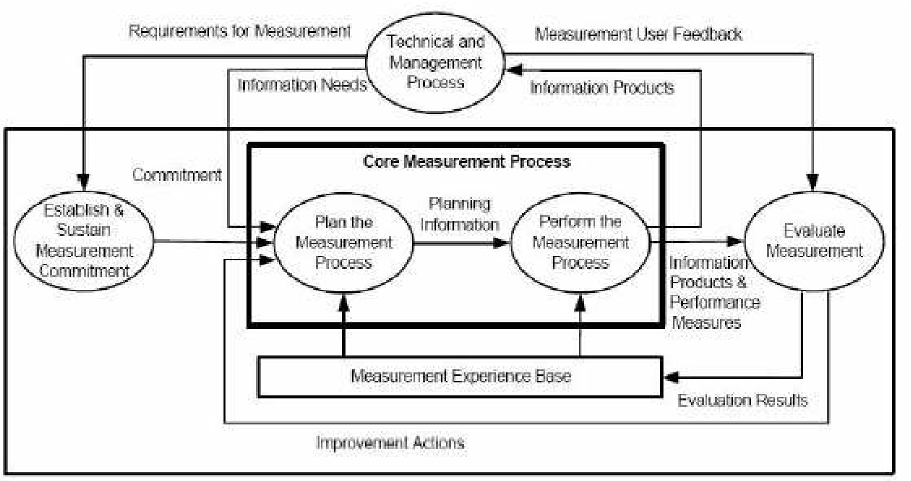

Requirements for Measurement
Technical and Management Process
Measurement User Feedback
Information Needs
Information Products
Commitment
Core Measurement Process
Establish & Sustain Measurement Commitment
Plan the Measurement Process
Planning Information
Perform the Measurement Process
Evaluate Measurement
Information Products & Performance Measures
Measurement Experience Base
Evaluation Results
Improvement Actions

Fig. 1. Outline of ISO/IEC 15939 Standard

According to Dumke [2] and Fenton [3], software measurement is directed to three main components of software development; products, processes and resources. But overwhelmingly, the considered activities of measurement in the IT quality assurance are mainly focused to the software product. The software process and their involved resources are the other important indicators. The software process resources could be divided into CASE (computer aided software engineering) tools as a support for the development and COTS (commercial off-the-shelf software) as external components and a part of the developed software product.

Software measurement web services when exploited can provide flexibility and ease of use probably not achievable through traditional measurement tools. Such a service can be availed to measure and analyze general software products, specifically open source components and COTS resources. COTS resources have an essential influence in the developed software product (see [4]). So we can establish a very great influence of the quality of chosen Java library (as a special kind of COTS) to the quality of the developed Java application (as the developed software product or system).

Furthermore, the quality of software development including their resources can be characterized through their maturity level. Especially the CMMI level four defines a quantitative process management. The measurement orientation and predefined metrics are described in [5]. This intention is also one of the reasons for involvement of software resources as COTS in the quality measurement process.

2 Applying SOA to software measurement

Being in most instances embedded in corporate measurement programs and/or quality assurance initiatives, multi-step measurement automation turns out to be a key factor for cost effectiveness and developer's commitment to the measurement process [6]. To extract, which measurement phases could actually benefit from automation endeavors and could thus be implemented by a service provider, we avail ourselves of Fenton and Pfleeger's [7] high-level measurement context as illustrated in figure 2 that alludes four aspects of automation and/or service areas: measurement itself, measurement collection, its collation and storage, and analysis of metrics results.

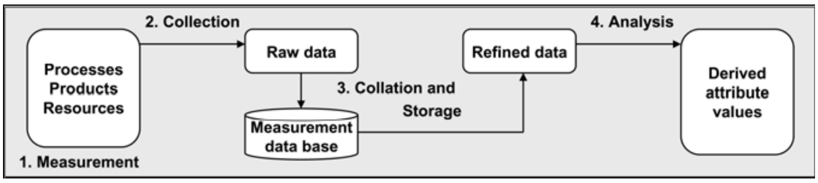


Fig. 2. Software Measurement Context

In figure 3 we present a dynamic composition of software measurement services. That procedure as illustrated in figure 3 would enable a service consumer to flexibly tailor a measurement environment at runtime according to its instantaneous needs in terms of the four service areas previously mentioned and cover all aspects by leasing services with SOA characteristics from a service provider via a registry as long as needed and combining them with the aid of a contract in contrast to investing in inappropriate and expensive conventional, hard-wired measurement software suites.

Examples of SOA-based measurements can be found in [8], [9], [10] and [11] and have been summarized in Fig 4.

Short descriptions of quality measurement services seen in figure 4 are given below (see also www.smlab.de and www.software-kompentenz.de).

a) CMMI evaluation for software process measurement and improvement
b) Execution of the COSMIC FFP (Full Function Points) for size estimation
c) Continuous evaluation of the performance of chosen (real) web services
d) SML@b experience factory for software metrics and measurement
e) (German) experience basis for software engineering (including software quality and quality improvement)

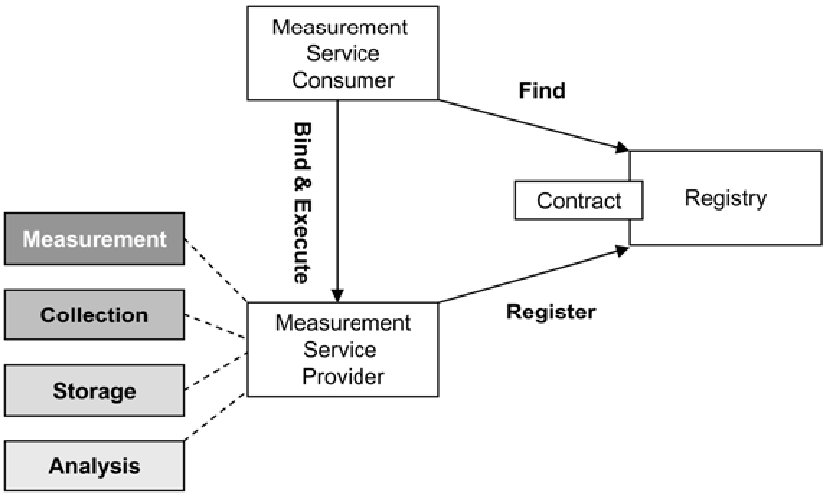


Fig. 3. Service-oriented software measurement

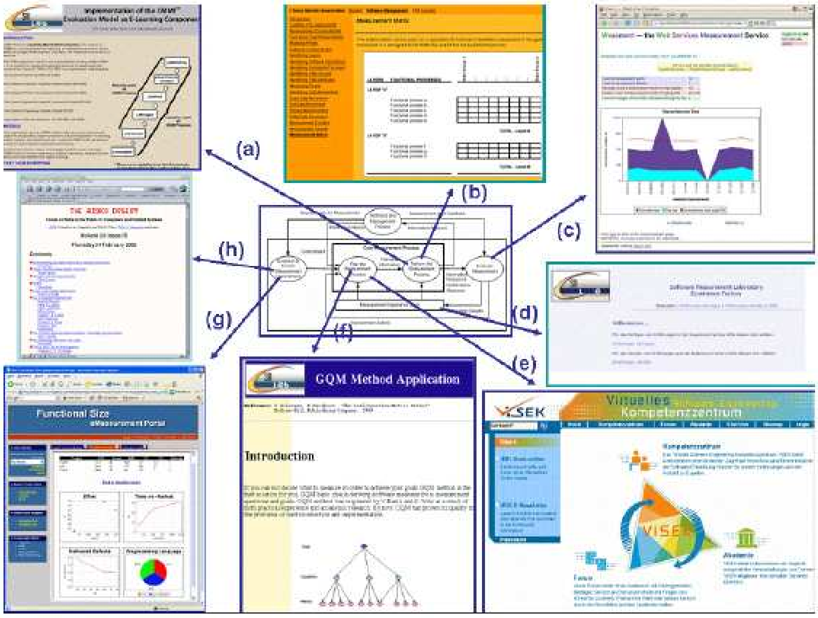


Fig. 4. Software quality and measurement web services

f) Goal question metrics (GQM) method description and application for measurement planning
g) SML@b's general overview about function size measurement (FSM)
h) Peter Neumann's Web page for software risks for quality motivation

3 Case study: An example measurement web service

In this section we present a sample software measurement web service and a web application as a case study. The **O**bject-**O**riented **M**easurements of **J**ava Technologies (OOMJ) [12] is a measurement web service in line with the SOA for software measurement discussed in section 2 (OOMJ can be accessed via www.smlab.de). OOMJ is aimed at providing an approach to evaluate Java components. The metrics evaluations supported by this service are:

- **Size metrics**: number of classes, number of methods, number of lines of code (Loc), average Loc per class, average methods per class
- **Chidamber & Kemerer metrics** [13]: number of children (NOC), depth of inheritance tree (DIT), weighted method per class (WMC), coupling between objects (CBO), response for a class (RFC), and lack of cohesion between methods (LCOM)
- **Abreu's MOOD metrics** [14]: method hiding factor (MHF), attribute hiding factor (AHF), method inheritance factor (MIF), attribute inheritance factor (AIF), and polymorphism factor (POF)

In addition to providing measurement and analysis of user projects, this service also acts as a repository containing measurement results for a wide variety of Java technology libraries such as J2SE 1.5.0, J2EE 1.2.1, J2EE 1.3.1, J2EE 1.4.1, J2ME, Javacard 2.2.1, JWSDP 1.5, Jakarta Tomcat 4.1.31, Jakarta Tomcat 5.5.9, DOM4J, some Apache projects, and few other open source Java-based projects.

4 Highlighting quality aspects of software products using OOMJ

This section discusses some useful applications of this web service. The various possibilities are discussed below:

4.1 Initial quality aspects based on size metrics

A project's sheer size is a fountainhead of failure. The larger the project, the more complexity there is in both its static and dynamic elements [15]. Greater complexity increases the possibility of errors, since it becomes hard to understand all interacting parts of software and it is one of the main reasons of software failures. Figure 5 compares various libraries/projects based on some size measures. These size measures serve a quick impression of possible complexity issues of given projects. The figure shows comparatively large class size for the project named Weka (http://sourceforge.net/projects/weka/). Thus this project is relatively more complex and will need higher effort in understanding its design aspects etc.

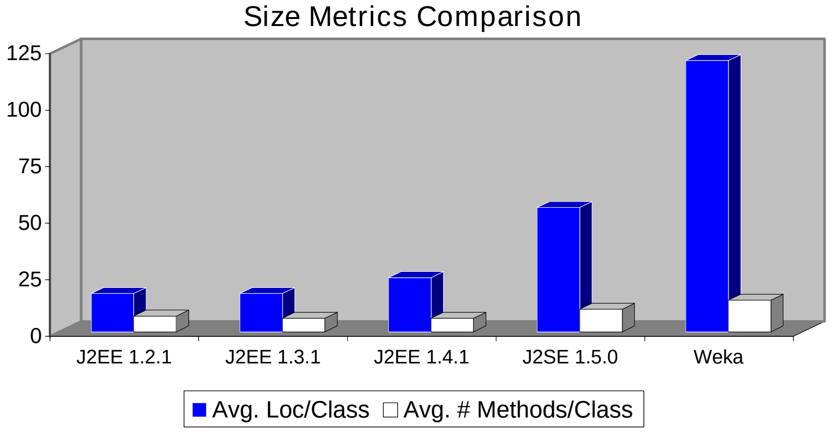


Fig. 5. Size metrics comparison of Java libraries/projects

4.2 Complexity Evaluation

There have been many empirical validations (of metrics supported by this web service) which have investigated a possible relationship between value of the metric and the complexity or other internal quality attribute of the underlying code or software design. For example, coupling between objects (CBO) affects change proneness [16] and higher inter-object class coupling calls for rigorous testing [17]. Therefore, a smaller CBO is advocated. Figure 6 shows average values of CBO for various Java libraries and some open source projects. CBO has been reported to be close to 3 for most Java libraries [12] . The graph shows a bit higher coupling level in case of selected Apache projects, Tomcat 5.5.9 and Weka project. These projects can be regarded relatively more complex, difficult to maintain and requiring more testing effort. Such comparisons based on other metrics, as well, for some projects or libraries can highlight critical design aspects which can attract more attention and revision thereby avoiding possible internal quality issues.

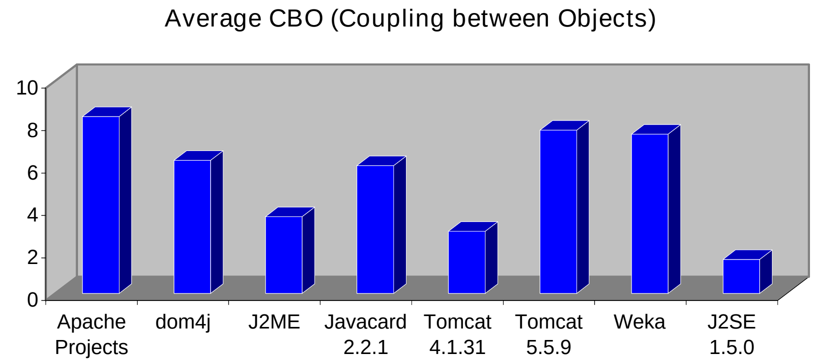


Fig. 6. Average of coupling measure for various Java libraries

4.3 Threshold analysis

According to Ebert et al [9] a step in planning a measurement process is measurement adjustment which involves determination of the favorable values and tuning of the thresholds as approximation during the software development from other project components. Here we present few examples using threshold analysis for the concerned metrics values. Farooq [12] provides empirical results for Chidamber & Kemerer and MOOD metrics based on a wide variety of Java standard libraries evaluations. Assuming the metrics values provided by him as desirable metrics ranges exercised by industry, we can perform a comparative analysis of the libraries and software of our interest to spot a deviation from a common industry practice.

For example, attribute inheritance factor (AIF, parts of MOOD metrics suite) has a typical value in the range of 20 to 30% [12]. Figure 7 shows value of this metric for different packages from J2SE 1.5.0. The figure highlights two packages which have comparatively higher value of this metric indicating a relatively higher use of attribute inheritance.

Figure 8 shows value of polymorphism factor (POF, another metric from MOOD). In comparison to most common range for POF quoted in [12] (which is close to 5%), this figure shows an unusually high use of polymorphism in

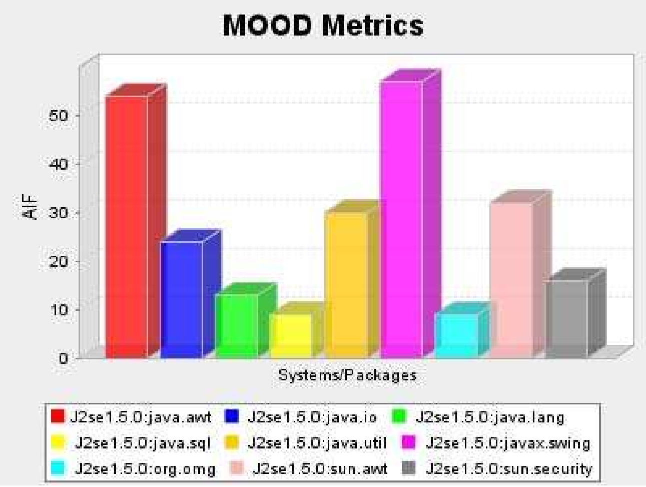


Fig. 7. Attribute inheritance factor (AIF) for few J2SE 1.5.0 packages

java.io package from J2SE 1.5.0 which may cause an increased complexity for this package.

4.4 Benchmarking

Benchmarking is used internally to a company for comparing projects and externally for comparing best practices and is of particular importance for the investigation and resulting application of best practices [9], [18]. COTs components can be benchmarked with the extensive measurement results available in OOMJ to analyze them from different aspects. Figure 9 presents one such example where an open-source library JFreeChart 1.0.0 is being benchmarked with J2SE 1.5.0 for coupling between objects (CBO metric from Chidamber & Kemerer metrics suite). Coupling between objects affects changeability of classes [17]. Therefore, a smaller CBO value is advocated. The graph shows there are more classes in JFreeChart 1.0.0 with higher coupling as compared with *java.awt* package from J2SE 1.5.0. Many other such comparisons can highlight important design characteristics indicating a need for design revision to keep it line with experience-based popular approaches reflected in the design of most standard libraries.

5 Conclusions and Future Work

After briefly discussing the general aspects of quality measurement, evaluation, and assurance in information technology, we propose applying the service-

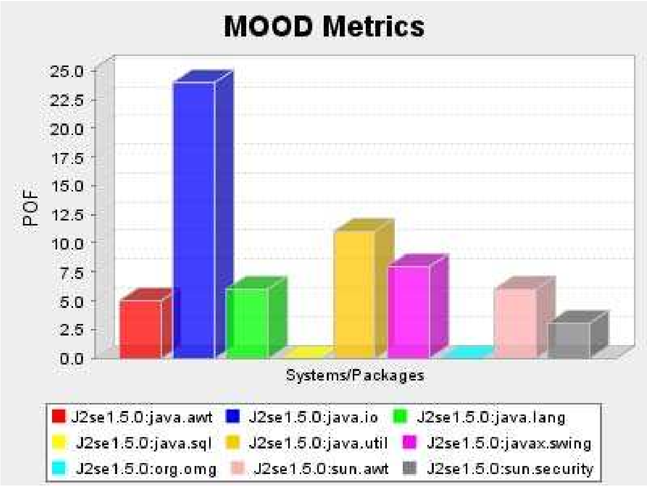


Fig. 8. Polymorphism factor (POF) for few J2SE 1.5.0 packages

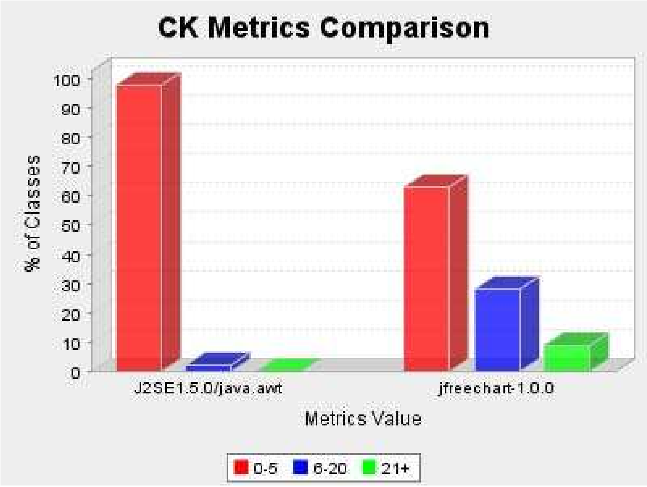


Fig. 9. Coupling between objects (CBO) for few related libraries

oriented approach towards quality assurance of software products. This paper introduces a composition of software measurement web services to derive internal quality indicators of Java software components using two sets of widely used object-oriented software metrics. Application of a sample software measurement web service, as a case study, is discussed exemplifying various ways in which this service can be beneficial in highlighting different quality aspects of standard Java libraries and other Java software components. In doing so, our proposed service-oriented measurement approach turns out to be an interesting and further to explore alternative to conventional software measurement suites containing merely a hard-wired set of functionalities.

The presented measurement service has focused on complexity attribute of software components. In future, this service could be enhanced by providing wider empirical experiences and could be enabled to apply other metrics values for general software development aspects like reliability, maintainability and stability etc. Additionally, rather than incorporating and providing a combined services approach, another extension to our sample web service could be provision of separate web services for all four measurement service areas discussed in section 2 .

References

1. *ISO/IEC 15939- Information Technology - Software Engineering -Software Measurement Process.* 2001.
2. Reiner. R. Dumke and E. Foltin. Metrics-based evaluation of object-oriented software development methods. In *CSMR '98: Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering*, page 193, Washington, DC, USA, 1998. IEEE Computer Society.
3. Norman E. Fenton. Software measurement: Why a formal approach? In *Proceedings of the BCS-FACS Workshop on Formal Aspects of Measurement*, pages 3–27, London, UK, 1992. Springer-Verlag.
4. Victor R. Basili and Barry Boehm. Cots-based systems top 10 list. *Computer*, 34(5):91–93, 2001.
5. Reiner R. Dumke, I. Cotè, and O. Andruschak. Statistical process control (spc) - a metrics-based point of view of software processes achieving the cmmi level four. Technical report, University of Magdeburg, Department of Computer Science, Magdeburg, Germany, 2004.
6. Tracy Hall and Norman Fenton. Implementing effective software metrics programs. *IEEE Softw.*, 14(2):55–65, 1997.
7. N. E. Fenton and S. L. Pfleeger. *Software Metrics: A Rigorous and Practical Approach.* International Thomson Computer Press, Boston, MA, USA, 1996.
8. M. Lother and Reiner R. Dumke. Application of emeasurement in software development. In *Proceedings of the IFPUG Annual Conference*, San Antonio, Texas, 2002.

9. C. Ebert, Reiner R. Dumke, M. Bundschuh, and A. Schmietendorf. *Best Practices in Software Measurement.* SpringerVerlag, 2004.
10. A. Schmietendorf and Reiner R. Dumke. A measurement service for monitoring the quality behaviour of web services offered within the internet. In *Proceedings of the IWSM/MetriKon 2004*, pages 381–390. Shaker Publ., 2004.
11. A. Farooq, R. Braungarten, and Reiner R. Dumke. An empirical analysis of object-oriented metrics for java technologies. In *INMIC 2005: Proceedings of the 9th IEEE International Multi-topic Conference*, December 2005.
12. A. Farooq. Conception and prototypical implementation of a web service as an empirical-based consulting about java technologies. Master's thesis, University of Magdeburg, November 2005.
13. S. R. Chidamber and C. F. Kemerer. A metrics suite for object oriented design. *IEEE Trans. Softw. Eng.*, 20(6):476–493, 1994.
14. F. B. Abreu and W. Melo. Evaluating the impact of object-oriented design on software quality. In *METRICS '96: Proceedings of the 3rd International Symposium on Software Metrics*, page 90, Washington, DC, USA, 1996. IEEE Computer Society.
15. Robert N. Charette. Why software fails. *IEEE Spectrum*, pages 36–43, September 2005.
16. G. Alkadi and D. L. Carver. Application of metrics to object-oriented designs,. In *Proceedings of IEEE Aerospace Conference*, volume 4, pages 159–163, March 1998.
17. Audun Foyen. Dynamic coupling measurement for object-oriented software. *IEEE Trans. Softw. Eng.*, 30(8):491–506, 2004.
18. M. Lother, R. Braungarten, M. Kunz, and Reiner R. Dumke. The functional size emeasurement portal: A web-based approach for effort estimation, benchmarking and elearning. In *IWSM/MetriKon 2004: Proceedings of the International Workshop on Software Metrics and DASMA Software Metrik Kongress*, pages 381–390. Shaker Publ., November 2004.

Towards an Internal Numerical Taxonomy of Software Process Assessment Methods

Alberto Sampaio[1], Edwin Gray[2], Mário Martins[3], Isabel B. Sampaio[4]

[1] Departamento de Engenharia Informática and
Laboris (Research Laboratory on Systems)
Instituto Superior de Engenharia do Porto, 4200-072 Porto, Portugal.
as@dei.isep.ipp.pt
www.{laboris, dei}.isep.ipp.pt
[2] School of Computing and Mathematical Sciences
Glasgow Caledonian University, Cowcaddens Road,GLASGOW G4 0BA, UK.
e.gray@gcal.ac.uk
www.gcal.ac.uk
[3] Departamento de Informática
Universidade do Minho, Campus de Gualtar, 4710-057 Braga, Portugal.
fmm@di.uminho.pt
www.di.uminho.pt
[4] Departamento de Engenharia Informática
Instituto Superior de Engenharia do Porto, 4200-072 Porto, Portugal.
is@dei.isep.ipp.pt
www.dei.isep.ipp.pt

Abstract. *Objectives*: The main goal of this paper is the internal comparison and classification of software process assessment methods. *Methods*: This exploratory study used a new methodology, based on numerical taxonomy. Eight software process assessment methods (SCAMPI; RAPID; SPICE; EVALUATION; BOOTSTRAP; SA-SI; MMA, and CBA-IPI) were classified. For the study were used documents describing the assessment methods. *Results*: A data table with a final set of 112 characters; proximity matrices of methods from step 4, and the final result was a hierarchical taxonomy of the eight methods selected from a set of taxonomies (step 5) validated (step 6) for a significance level of 5%. Conclusions: The internals comparison and classification produced are preliminary and are in accordance with our perceptions about the internal relationships between the methods. From the taxonomy was concluded that the methods SCAMPI and Bootstrap are the closest, followed by CBA-IPI, RAPID and MMA, Evaluation, SA-SI and SPICE by this order. Because of the exploratory nature of the work, the need for more studies is pointed.

1 Introduction

Software process assessment (SPA) is an important step in the software process improvement (SPI) cycle. Because SPA results are the input for the SPI initiatives, a “good” assessment can be seen as one that allows a successful improvement. To know the reasons why an assessment method contributes to such a “good” assessment it is necessary to know which attributes, and their combinations, that successful methods possess and that the others do not. So, it is necessary to compare current methods about their internal characteristics, as done in this paper. Such comparisons, together with empirical data about the use of the methods, will allow the discovery of which characteristics should be present in “good” SPA methods.

Software process improvement, in general, and the assessment area, in particular, are far from being guided by a scientific theory and could not be considered scientifically mature. In a comparison or classification of SPA methods, if it is lacking a theoretical base, then it is not rational to prefer some characteristics in detriment of others. This implies the need for a comparison and classification method where all known characteristics could be considered.

Some comparisons and classifications have been made about strategies and models of process improvement, but very few of SPA methods are described in the literature. Additionally there are no well defined methods for comparing SPA methods, as there is a lack of approaches for software methods in general [39]. Those methods belong, in general, to the second and fourth classes of Henk Sol [38] types of methods of comparison. The goal of our work is the comparison and classification of SPA methods based on their internal characteristics. The comparison and classification was done with a new methodology based on numerical taxonomy, as proposed initially for biological sciences, which solves the previous limitations.

In the next sections we present related work, the research approach and its application, after which we discuss results and present some conclusions.

2. Background and Related Work

It is possible to find a reasonable number of studies comparing SPA models, as [2], [13], [14], [22], [24], [25], [26], [33], [41], [42], [43] [44], [47], but only a reduced number of comparisons of SPA methods. The comparisons of SPA methods we could find reported in the literature are, Ares et al. [1], Haase [12], Rout[28], Rout and Gasston [29], Rout and Nielson [30], Wang et al. [45], Zahran [47], and the Software Engineering Institute (SEI) [36].

However there are no simultaneous studies of comparison and classification and none of the studies could be considered adequate: usually the studies analyze only a few methods or a few characteristics depending of each author’s criteria or goals and without a well defined comparison process, with all the consequent problems, like lack of uniformity, difficult to determine its quality, difficult of construction and impossibility to replicate the classification. Replication studies are considered important in an empirical software engineering but, usually, are not developed [3].

It was necessary to use a new approach for the comparison and classification of software methods, and SPA methods in particular. The new methodology used allows the creation of a numerical taxonomy of the methods. Because it analyzes methods, the approach is referred as a methodology. The approach is considered empirical, statistical and multidimensional. It is based on a numerical taxonomy used in biology that was, mainly proposed in the end of the fifty's by Sokal and Sneath [37] to classify live beings. Despite its origin, the numerical taxonomic methods are considered applicable to other areas [20] [15].

The methodology advocates the importance of gathering as many characteristics as possible about each entity (method) to be classified in order to produce a “natural” classification. This is known as the saturation principle and that is consistent with the lack of a theoretical base as stated previously. Because it is a rigorous and numerical approach, studies that use it could be more easily replicated.

3. The Comparison and Classification Methodology

In the methodology the resulting classification is hierarchical. Each entity is designated by OTU (“Operational Taxonomic Unit”) and the characteristics by characters. The methodology comprises the following steps:

1. Choose the OTU’s – In this step the SPA are chosen.
2. Character discovery and measurement - Consists of determining which characters should be included and the “measurement" for each OTU.
3. Codification – Codification of the characters and their normalization; It also involves the treatment of missing data.
4. Proximity – In this step the proximity matrix is produced showing the dissimilarities between the OTU’s. This gives a value of “global affinity” between the methods. In the calculation several coefficients of dissimilarity are used.
5. Clustering – This step try to group the methods. Several clustering criteria are applied in a hierarchical agglomerative strategy. This is the cluster analysis step.
6. Validation - Statistical validation of the results in order to find the best solution. Validation is done based on the idea of stability.
7. Interpretation – Interpretation of the solution – It includes the formation of potential groups and conceptualization. This step requires knowledge of the application domain of the study (SPA methods in the present case).
8. a) Data extraction - given a taxonomic group, obtain the data about that group; b) Identification of cases - determination of the respective group of a given method.

The first three steps allow the creation of a data table with the characters for each method, which is the usual result of comparative studies. With this methodology we can go a step further by allowing the development of a classification of the entities. With the methodology interpretation is possible at the various steps, but it is done mainly at the seventh. The authors of the methodology advocate the application of several measures at step 4, and of several methods (criteria) at step 5, in order to produce several classifications. The consistency between the several results is considered an indicator of the quality of the solution. The classification with the smallest error could be selected as the solution.

4. The Study

The study is exploratory and followed the methodology steps, rules and recommendations. These rules and recommendations affected the decisions. Step number seven is not covered because it was not completely developed and because it consists mainly of numbers identifying the characters belonging to each of the classes found making no sense without the data table. The last step is not part of the study.

Given the high quantity of numerical computation required, the Matlab software package was used. Notwithstanding the enormous functionality offered by the package, notably from its “statistics toolbox”, it was necessary to develop many new Matlab functions using its built-in language. Routines were developed for numeric processing at steps, 3, 4, 5, 6 and 7. Computation was done in a 1GHz laptop with 2*256 MB of main memory and Microsoft Windows XP operating system.

4.1. Choose the OTU’s (Step 1)

The study attempted to include all the SPA methods that have some documentation describing them. However it was not possible to include all the SPA methods, because there was not enough information available for some of them. In some cases the methods were not freely available. Older versions of existing methods have not been included. Because methods using only a questionnaire are considered too limited [11] they have not been included. There was also a set of methods planned to be included later in the classification (step 8) and that will allow test classification sensibility to new cases. Eight methods were chosen. From a sampling theory point of view it is an intentional non-probabilistic sample. The methods selected and their main reference sources are listed in Table 1. The set of methods has dimension *n*=8.

Table 2. Methods included and their sources.

Method	Author	Main sources
1. SCAMPI	SEI	[35],[36]
2. RAPID	T. Rout et al.	[31]
3. 15504	ISO/IEC	[10],[19]
4. Evaluation	J. Ares et al.	[1]
5. Bootstrap 3	Bootstrap Institute	[4],[5],[40]
6. SA-SI	M. Hobday, T. Brady	[17]
7. MMA	Karl Wiegers, D.Sturzenberger	[46]
8. CBA-IPI	SEI	[9],[21]

4.2. Character Discovery and Measurement (Step 2)

Character discovery was done OTU by OTU, by reviewing exhaustively all the documents related to the methods in order to find items that could be recognized as characteristics of a method. The discovery began with the SCAMPI method, because it was one of the biggest, this is, more documented. This was easy because SCAMPI requirements were all well stated in a document [36] from SEI. Our list of characters reflects that. The set of characters about the ISO/IEC 15504, was obtained through a table [36], connecting the SCAMPI method with 15504, and with the ISO/IEC15504 documentation. Because character discovery is empirical, each character being discovered by observation of its method there are no empty variables. A total of 165 characters were discovered.

For the cases where the character presents some well-defined values, these values were used. If not, a dichotomous classification scale, like "Present" and "Absent" was adopted by default. Some times the two state scale gave place to an ordinal scale of the kind, "Absent", "Partially Present" and "Present". The scale in these cases is ordinal. In other situations there was the need to use nominal multi-state and quantitative data. With the discovery and measurement of the characters a data table[21] with the 165 characters was constructed. An extract of three characters of the data table is presented in Table 2, where "S" means "Present" and "N" "Absent".

Table 3. An example of three characters.

Nº	Characters \ Methods	1	2	3	4	5	6	7	8
1	The Sponsor is a Senior manager	S	S	S	N	S	S	N	S
2	The Sponsor is a manager, but not Senior	N	N	N	S	N	N	N	N
3	The Sponsor is from the SPEG	N	N	N	N	N	N	S	N

4.3. Data Codification (Step 3)

The data were codified with the integer values {0,1} for the {S,N} case, to {0,1,2} for the {S,P,N} case, and no change was made for quantitative values. Next it was standardized in amplitude to the interval 0 to 1. There was a significant number of missing data and it did not make sense to apply a statistical procedure to treat the missing data. Note that there is about 1/20 ratio of entities by variable, but regression methods usually require much more cases than variables. So, it was applied the simplest solution, variable-deletewise, that means to delete the variables with missing data. After the deletion a total of p=112 characters remained.

4.4 Proximity (Step 4)

In this step the dissimilarity matrices were calculated by applying 5 coefficients (Euclidean, Standardized Euclidean, City Block, Minkowski with p=0.5 and Minkowski with p=4). These distances assume that data is quantitative. The resulting proximity matrices are shown in the Appendix. Each entry represents a distance between a pair of methods, in a total of $N = n\ (n\text{-}1)\ /\ 2 = 28$ distances.

[21] The table, in Portuguese, is available from the first author.

4.5 Clustering (Step 5)

For clustering, four hierarchical criteria (Single Linkage; Complete Linkage; Average; and Ward) were applied. These criteria could produce very different classifications and induce ultrametrics. The result was a total of twenty classifications. The classifications are presented in the form of indexed trees, called dendrograms. The index is a function describing the strength of the connection between nodes.

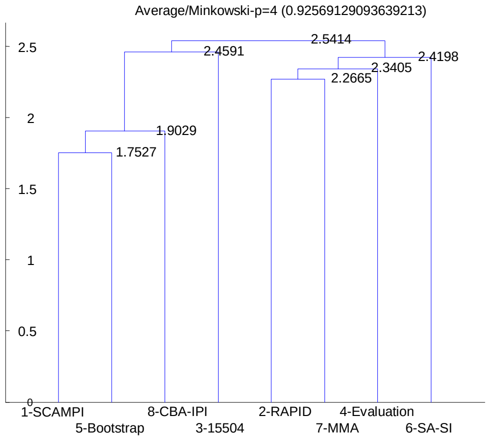


Fig. 3. The dendrogram of classification number 19.

4.6. Validation (Step 6)

The goal is to find the classification with the small error introduced during the classification, defined by the distortion introduced by the clustering method. The statistic chosen was the coefficient of cophenetic correlation (CCC) which is complementary to the clustering error. In Fig.1 the value between parentheses is the CCC. The significance of the results obtained was tested for each method for a significance value of 0.05. Because the CCC distribution depends from application to application, the null distribution for each method was determined by Monte Carlo sampling, with a number of 1000 samples. From the twenty initial classifications, five were rejected.

The concordance between the dendrograms of the statistical significant classifications was analyzed by applying the Kendall's W coefficient of concordance to their topologies (described by the Partition Membership Divergence [27] topological descriptor). The value obtained confirmed concordance for a significance level of 0.01. To select the classification with the smallest error it was necessary to select the classification with the highest CCC. Such solution is the number 19, Average/Minkowski-p=4, which is showed in Figure 1. But the solution number 3, Average/Euclid, has nearly the same value (equal to the third decimal place).

5. Discussion and Validity of Results

Methods SCAMPI and Bootstrap from the solution were the first to be joined, followed by CBA-IPI. Next RAPID and MMA were joined, followed by Evaluation and SA-S.I In the end SPICE joined the first group. The proximity of Bootstrap and SCAMPI is understandable because both methods were developed from a previous SEI method and from the SEI CMM model. They are also ISO/IEC 15504 compliant. CBA-IPI is also a SEI method, and SCAMPI was directly based on it. Methods RAPID and MMA are the most light. SA-SI is more problem oriented and depends only partially on an assessment model (the CMM). The Evaluation method is the only one based on a theory (theory of evaluation). SPICE is a standard and so it possess the minimum set of requirements that methods should have to be SPICE compliant. This could explain the distance of SPICE.

Some results obtained from the matrices followed by possible explanations:

1. In the dissimilarity matrices the most similar methods are 1 and 5, SCAMPI and Bootstrap, or 1 and 8, SCAMPI and CBA-IPI. Between the more dissimilar methods the variability is higher.
2. In the matrices, CBA-IPI appears closer of SCAMPI than of Bootstrap.
3. SPICE appears nearly equidistant of SCAMPI and Bootstrap.
4. The methods MMA e RAPID appears always near each other.

Possible explanations, in the same order:

1. These proximities were expected as explained earlier.
2. Both SACMPI and CBA-IPI were developed by SEI, but Bootstrap was not.
3. Bootstrap is compliant with 15504, but much more complete. SCAMPI uses CMMI models but it can be compliant with the standard if used with the continuous version of the CMMI.
4. Again, these two methods are less heavy than the others. However RAPID is SPICE based but MMA is CMM oriented. MMA is also more flexible.

Some validity issues should be pointed. The main limitations of the study are of two kinds: limitations with the numerical taxonomy itself; and construction of the data table by one person only, the first author. Character discovery is a subjective task based on the analyst intuition and knowledge of the domain. However, was possibly compensated by the large number of characters as advocated by the methodology. Because character discovery was done by only one person we couldn’t evaluate the validity of that process. The differences in the amount and quality of information between the methods documentation could also have affected the results. The character discovery and measurement started with the SCAMPI method described in the document ARC-Assessment Requirements for CMMI [36] and its use probably introduced some bias in the character discover process. Because several requirements have been divided, some deleted and new ones added, hopefully we reduced a possible bias. Finally, missing data was in a significant number. All this implied that the results could not be considered definitive.

3. Conclusions and Future Work

The goal of this study was the development of a comparison and classification of SPA methods based on their internal characteristics. It was used an innovative approach based in the numerical taxonomy as used in biology to classify live beings. An internal classification is in agreement with measurement theory because external measurement depends on the measurement of internal attributes. Direct measures occur in an initial phase of scientific development [48], as it is the case with SPA. As long as we know, this is the first numerical comparison and classification of SPA methods, or of software methods of any kind based on their internal characteristics.

The classificatory study was two-fold exploratory. From one side some SPA methods have never been compared and some characteristics never included, from other side, it was the first time this new comparison and classification methodology has been used. In the study eight SPA methods were compared and classified. The classification was done based on 112 characters. The chosen solution was the hierarchical taxonomy number 19. Potential groups were not created because of the exploratory nature of the study. However, it was apparent a group of three methods, SCAMPI, Bootstrap and CBA-IPI.

On this stage of work we were not motivated by practical reasons, however, with the actual results is possible to choose a method in a specific situation by being aware of the relevant characters and their importance. With this information the user could consult the data table to know whose methods better satisfy such characters. Or, using information from step 7, he could inspect in the taxonomy the classes that possess the relevant characters.

A future review of the SPA methods could allow a reduction of missing data, and then, another strategy for missing data should be considered. The subjective task of character discovery also needs to be improved. The binary and ordinal scales were treated as interval scales, because such violations could be considered acceptable in an exploratory study [6] and we did not find that the effects of such violations did produce significant differences in the results. Obtaining more documentation for the study proved to be difficult.

This work is included in a more wide work intended to identify the relationships between the internal characters and the results produced by the methods (the external characteristics). This means to apply the methodology, or part of it, two more times, to detect external characters and to relate external with internal characters. So other studies should be developed. To use the methodology in such staged way means that it can be seen as an empirical research methodology that could allow detect cause-effect relations. About the methodology it is planned to complete step 7 with concept formation and the study of the validity of the saturation principle with software methods documentation. We also intend to apply the methodology to other kinds of software methods and software phenomena.

Acknowledgement

The authors would like to thank Barbara Kitchenham for the remarkable comments to a previous version of this paper contributing in a significant way for its improvement.

References

[1] Ares, J., Garcia, R., Juristo, N., López, M., Moreno, A, A More Rigorous and Comprehensive Approach to Software Process Assessment, Software Process: Improvement and Practice, John Wiley & Sons, Vol.5, I.1, 3-30, 2000.

[2] Bamford, R.C., Deibler II, W.J., Comparing, Contrasting ISO 9001 and the SEI Capability Model, IEEE Computer, Vol.26, N., 10, pp.68-70, October, 1993.

[3] Basili, V., Shul, F., Lanubile, F., Building Knowledge through Families of Experiments, IEEE Transactions on Software Engineering, pp.456-473, Vol.25, N.4, July/August, 1999.

[4] Bicego, A., Khurana, M., Kuvaja, P., Bootstrap 3.0 Software Process Assessment Methodology, In C. Hawkins , M. Ross , G. Staples (Editors), Software Quality Management VI: Quality Improvement Issues, Springer-Verlag, NY, 1998.

[5] Bootstrap, Bootstrap: Europ's Assessment Method, Software, Vol.10, N.3, May, 1993.

[6] Briand, L., El-Emam, K., Morasca, S., On the Application of Measurement Theory in Software Engineering, Empirical Software Engineering: An intl. Journal, Vol.1, N.1, 1996.

[7] CSTB - Computer Science and Technology Board (National Research Board from the National Academy of Sciences), Scaling Up: A Research Agenda for Software Engineering". CACM, Vol.33, N.3, pp.281-293, 190.

[8] Dion, Raymond, "Process Improvement and the Corporate Balance Sheet," IEEE Software, July 1993, pp. 28-35.

[9] Dunaway, D., Masters, S., CMM-Based Appraisal for Internal Process Improvement (CBA IPI) V1.1: Method Description, Software Eng. Institute, CMU/SEI-96-TR-007, 1996.

[10] El Amam, K., Drouin, J., Melo, W., (Editors), SPICE: The Theory and Practice of Software Process Improvement and Capability Determination", IEEE CS Press, 1998.

[11] Gray, E.M., Sampaio, A., Benediktsson, O., An Incremental Approach to Software Process Assessment and Improvement, Proceedings of the British Computer Society Quality Special Interest Group's 11th Annual International Software Quality Management Conference: 'SQM 2003', 23rd - 25th of April, 2003, Glasgow Caledonian University, Glasgow, Scotland, 2003.

[12] Haase, V. H., Software process assessment concepts. Journal of Systems Architecture, the EUROMICRO Journal, Vol.42, N. 9, pp 621-631, Dec., 1996.

[13] Hailey, V.A., A Comparison of ISO9001 and the SPICE Framework, In K. El Amam, J. Drouin, W. Melo, (Editors), SPICE: The Theory and Practice of Software Process Improvement and Capability Determination", IEEE CS Press, 1998.

[14] Halvorsen, C.P., Conradi, R., A Taxonomy of SPI Frameworks, Twenty-Fourth Annual Software Engineering Workshop, SEL/NASA/GSFC, 1-2 December 1999.

[15] Hartigan, J.A., Classification, In Kotz, S., Johnson, N.L. (Editors-in-Chief), Encyclopedia of Statistical Sciences, (pp.1-10), Vol.2, John Wiley & Sons, Inc., 1982.

[16] Hetzel, B., The Sorry State of Software Practice Measurement and Evaluation, in Fenton, N., Whitty, R., Iizuka, Y. (Eds.), Software Quality Assurance and Measurement: A Worldwide Perspective, Thomson Comp. Press, 1995.

[17] Hobday, M., Brady, T., A Fast Method for Analysing and Improving Complex Software Processes, R & D Management, Vol.30, N.1, (pp.1-21), Blackwel Publishers Ltd, 2000.

[18] Humphrey, W.S., T.R. Snyder, and R.R. Willis, "Software Process Improvement at Hughes Aircraft," IEEE Software, July 1991, pp. 11-23.

[19] ISO/IEC TR 15504 - Information technology - Software process assessment — Parts 3,4,5, 1998.

[20] Jardine,N., Sibson, R., Mathematical Taxonomy, Wiley Series in Probability and Mathematical Statistics, John Wiley & Sons, 1971.

[21] Masters, S., Bothwell, C., CMM Appraisal Framework, Version 1.0, Software Engineering Institute, CMU/SEI-95-TR-001, 1995.

[22] Messnarz, R., A Comparison of BOOTSTRAP and SPICE, IEEE Software Process Newsletter, N.8, Winter, 1997.

[23] Osterweil, L.J., Software Processes Are Software Too, Revisited: An Invited Talk on the Most Influencial Paper of ICSE 9, pp.540-548, Proceedings of the 19th International Conference on Software Engineering, May 17-23, 1997, Boston, USA, ACM, 1997.

[24] Paulk, M.C., Comparison of ISO 9001 and the Capability Maturity Model for Software, Software Engineering Institute, CMU/SEI-94-TR-012, 1994.

[25] Paulk, M.C., Analyzing the Conceptual Relationship Between ISO/IEC 15504 (Software Process Assessment) and the Capability Maturity Model for Software, 1999 International Conference on Software Quality Cambridge, MA, 1999.

[26] Paulk, M.C., Konrad, M.D., Garcia, S.M., “CMM Versus SPICE Architectures,” IEEE Software Process Newsletter, No. 3, Spring, 1995a.

[27] Podani, J., Dickinson, T.A., Comparison of Dendrograms: a multivariate approach, Canadian Journal of Botany, Vol.62, pp.2765-2778, 1984.

[28] Rout, T.P., SPICE and the CMM: Is the CMM Compatible with ISO/IEC 15504?, AquIS ’98, Venice, Italy, March, 1998.

[29] Rout, T.P., Gasston, J.L., Different Approaches to Software Assessment, In T.P. Rout (Ed.), Software Process Assessment and Improvement, Computational Mechanics Pub., 1998.

[30] Rout, T.P., Neilson, M.P., Gasston, J.L., Experiences with the use of different approaches to software process assessment, Proceedings of the 2nd International Conference on Software Quality Management. Edinburgh. Scotland, 1994.

[31] Rout, T.P., Tuffley, A., Cahill, B., Hodgen, B., The Rapid Assessment of Software Process Capability, Proceedings of SPICE 2000 Intl. Conference, Limerick, Ireland, 2000.

[32] Rozum, J., Concepts on Measuring the Benefits of Software Process Improvements, Software Engineering Institute, CMU/SEI-93-TR-009, 1993.

[33] Saiedian e McClanahan, Frameworks for Quality Software Process: SEI Capability Maturity Model versus ISO 9000, Software Quality Journal, Vol.5, pp.1-23, 1996.

[34] Sampaio, A., Gray, E.M., Martins, M., A Comparison of SPA Methods, 31ts EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA), WiP session , August 31st-September 3rd, 2005, Porto, Portugal, 2005.

[35] SEI (CMMI Product Development Team), SCAMPI(SM), V1.0 Standard CMMI SM Assessment Method for Process Improvement: Method Description, Version 1.0, Technical Report CMU/SEI-2000-TR-009, 2000.

[36] SEI (CMMI Product Development Team), ARC, V1.0 Assessment Requirements for CMMI, Version 1.0, Technical Report CMU/SEI-2000-TR-011, 2000.

[37] Sokal, R.R., Sneath, P.H.A, Principles of Numerical Taxonomy, W. H. Freeman and Company, 1963.

[38] Sol, H.G., A Feature Analysis of Information Systems Design Methodologies: Methodological Considerations, pp.1-7, in T.W. Olle, H.G. Sol, C. Tully (Editors), Information Systems Design Methodologies: A Feature Analysis, IFIP WG8.1 Working Conference, 5-7-July, 1983, UK, North-Holland, 1983.

[39] Song, X., Osterweil, L.J., Toward Objective, Systematic Design-Method Comparisons, IEEE Software, Vol. 9, N.3, pp.43-53, May, 1992.

[40] Stienen, H., Engelman, F., Lebsanft, BOOTSTRAP: Five Years of Assessment Experience, In Proceedings of the 8th Int. Workshop on Software Technology and Engineering Practice (STEP'97), IEEE Press, 1997.

[41] Tingey, M.O., Comparing ISO 9000, Malcolm Baldridge, and the SEI CMM for Software: A Reference and Selection Guide, Upper Saddle River: Prentice-Hall, 1997.

[42] Tully, C., Kuvaja, P., Messnarz, R., Software Process Analysis and Improvement: a Catalogue and Comparison of Models, In Richard Messnarz, Colin Tully (Eds), Better Software Practice for Business Benefit, IEEE Computer Society, Los Alamitos, California, 1999.

[43] Wang, Y., Court, I., Ross, M., Staples, G., King, G., Dorling, A., Quantitative Analysis of Compatibility and Correlation of the Current SPA/SPI Models, In Proceedings of the 3rd Int. Software Engineering Stantards Symposium (ISESS'97), IEEE Press, 1997a.

[44] Wang, Y., Court, I., Ross, M., Staples, G., King, G., Dorling, A., Quantitative Evaluation of the SPICE, CMM, ISO 9000 and BOOTSTRAP, In Proceedings of the 3rd Int. Software Engineering Stantards Symposium (ISESS'97), IEEE Press, 1997b.

[45] Wang, Y., King, G., Dorling, A., Wickberg, H., King, G., Experience in Comparative Process Assessment with Multi-Process-Models, Proceedings of the 25th Euromicro Conference (EUROMICRO '99), IEEE Press, 1999c.

[46] Wiegers, K.E., Sturzenberger, D.C., A Modular Software Process Mini-Assessment Method, IEEE Software, Vol.17, N.1, 2000, pp.62-69.

[47] Zahran, S., Software Process Improvement: Practical Guidelines for Business Success, Addison-wesley, 1998.

[48] Zuse, H. A Framework of Software Measurement, Walter de Gruyter&C., Berlin, 1997.

Appendix: Proximity Matrices

Euclidean

6.94	5.79	6.38	3.28	7.09	7.06	3.28
	5.71	5.82	6.55	6.18	5.50	6.92
		5.89	5.50	6.69	7.03	6.38
			6.20	6.04	5.71	5.96
				6.89	6.83	4.30
					6.29	6.71
						6.64

CityBlock

```
51.34  36.00  42.50  12.50  54.50  55.83  11.50
       36.34  36.84  44.84  43.16  35.49  50.84
              37.50  31.50  50.50  54.83  43.50
                     41.00  41.00  38.33  38.00
                            50.00  53.33  21.00
                                   45.33  49.00
                                          49.33
```

Minkowski p=4

```
 2.60   2.37   2.51   1.75   2.62   2.60   1.79
        2.34   2.37   2.68   2.42   2.27   2.60
               2.39   2.51   2.52   2.59   2.49
                      2.46   2.40   2.31   2.41
                             2.74   2.54   2.02
                                    2.44   2.55
                                           2.52
```

Standardized Euclidean

```
 3.31   2.69   3.07   1.58   3.32   3.29   1.59
        2.53   2.67   3.24   2.68   2.47   3.29
               2.70   2.70   2.95   3.27   3.02
                      3.02   2.67   2.50   2.84
                             3.34   3.22   2.09
                                    2.74   3.09
                                           3.05
```

Minkowski p=0,5

```
2923.39  1449.41  1931.58   194.60  3366.42  3701.25  146.93
          1558.87  1549.31  2266.13  2232.32 1585.78 2847.43
                   1582.30 1129.40  3054.05  3531.18 2095.64
                           1855.12  2000.59  1860.27 1605.69
                                    2824.09  3476.19  532.27
                                             2513.10 2736.72
                                                     2885.47
```