Lecture Notes in Computer Science

The LNCS series reports state-of-the-art results in computer science research, development, and education, at a high level and in both printed and electronic form. Enjoying tight cooperation with the R&D community, with numerous individuals, as well as with prestigious organizations and societies, LNCS has grown into the most comprehensive computer science research forum available.

The scope of LNCS, including its subseries LNAI and LNBI, spans the whole range of computer science and information technology including interdisciplinary topics in a variety of application fields. The type of material published traditionally includes

- proceedings (published in time for the respective conference)
- post-proceedings (consisting of thoroughly revised final full papers)
- research monographs (which may be based on outstanding PhD work, research projects, technical reports, etc.)

More recently, several color-cover sublines have been added featuring, beyond a collection of papers, various added-value components; these sublines include

- tutorials (textbook-like monographs or collections of lectures given at advanced courses)
- state-of-the-art surveys (offering complete and mediated coverage of a topic)
- hot topics (introducing emergent topics to the broader community)

In parallel to the printed book, each new volume is published electronically in LNCS Online.

Detailed information on LNCS can be found at
www.springer.com/lncs

Proposals for publication should be sent to
LNCS Editorial, Tiergartenstr. 17, 69121 Heidelberg, Germany
E-mail: lncs@springer.com

ISSN 0302-9743
ISBN 978-3-540-85552-1
9 783540 855521

Lecture Notes in Computer Science
LNCS LNAI LNBI

Cuadrado-Gallego et al. (Eds.)

LNCS 4895

Software Process and Product Measurement

LNCS 4895

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

Software Process and Product Measurement

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

Lecture Notes in Computer Science 4895

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

Software Process and Product Measurement

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

Springer

Volume Editors

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

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

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

Library of Congress Control Number: 2008933374

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

LNCS Sublibrary: SL 2 – Programming and Software Engineering

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

Springer is a part of Springer Science+Business Media

springer.com

Printed in Germany

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

Preface

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

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

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

February 2007

Juan J. Cuadrado-Gallego

---

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

Organization

General Chairs

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

Program Committee Chairs

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

Program Committee

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

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

Special Sessions/Workshop Chairs

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

Table of Contents

Using Controlled Experiments for Validating UML Statechart Diagrams Measures

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

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

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

1 Introduction

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

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

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

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

J.J. Cuadrado-Gallego et al. (Eds.): IWSM-Mensura 2007, LNCS 4895, pp. 129–138, 2008.

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

2 Measures Definition

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

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

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

Our work has refined the referred method by formally defining the measures using two formal languages: OCL [17] and Maude [9]. Table 1 presents the informal definition of the measures in natural language whilst Fig. 1 provides an example of the formal definition of the measure NCS using OCL and Fig. 2 provides an example of the formal definition of the measure NIA using Maude.

Table 1. Definition of the measures in natural language

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

```
context StateMachine::NCS():Integer

body: result = self.top.allSubstates()->collect(s |
               s.oclIsTypeof(CompositeState))-> size()
```

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

```
...

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

...

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

...

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



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

...
```

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

3 Theoretical Validation

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

In our work, we have based on the property-based framework proposed by Briand [3, 4] and the Measurement Theory-based DISTANCE framework [18]. The first one has characterized the measures as size or complexity measures, while the second framework has characterized all the measures as ratio scale.

4 Empirical Validation

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

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

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

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

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

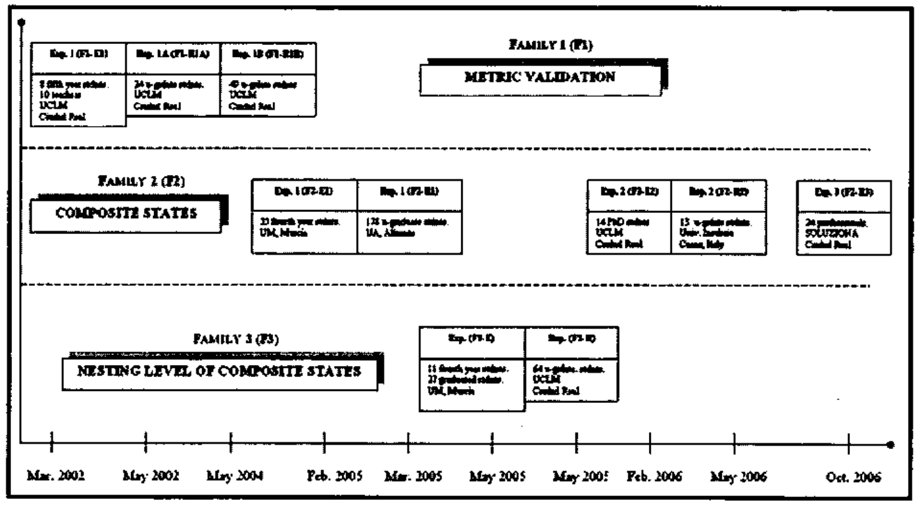


Fig. 3. Chronology of the families of experiments

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

This first family of experiments was performed for studying the relationship between the different defined measures and the understandability of UML statechart diagrams. The main characteristics of this family can be found in Table 2, while Fig.4 graphically shows the average values that were obtained for the different measurements related to the dependent variable.

Table 2. Characteristics of the family F1

Subjects	E1: 8 PhD students, 10 teachers R1A: 24 students R1B: 49 students
Location	University of Castilla-La Mancha (Spain)
Date	E1: March 2002 R1A: May 2002 R1B: May 2004
[illegible]	20 diagrams with different values for the measures
Dependent	Understandability of UML statechart diagrams, measured by UT (time), UCorr (correctness) and UCom (completeness), later by UEffic (efficiency).
Independent	Measures for UML statechart diagrams understandability
[illegible]	$H_{0,1}$: There is not a significant correlation between the UML statechart diagrams measures and the understandability time. $H_{0,2}$: There is not a significant correlation between the UML statechart diagrams size measures and understandability correctness. $H_{0,3}$: There is not a significant correlation between the UML statechart diagrams size measures and understandability completeness.

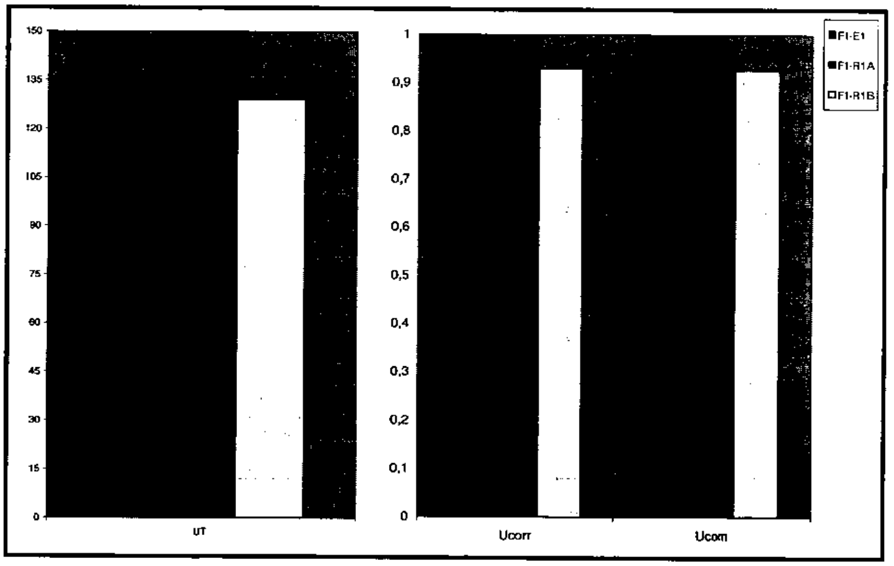


Fig. 4. Average values for UT, UCorr and UCom in F1

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

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

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

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

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

$$UEff = 0.011575 - 0.000813*NA - 0.000204*SSF - 0.000273*AWS \quad (1)$$

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

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

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

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

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

Table 3. Characteristics of the family F2

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

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

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

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

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

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

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

Table 4. Characteristics of the family F3

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

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

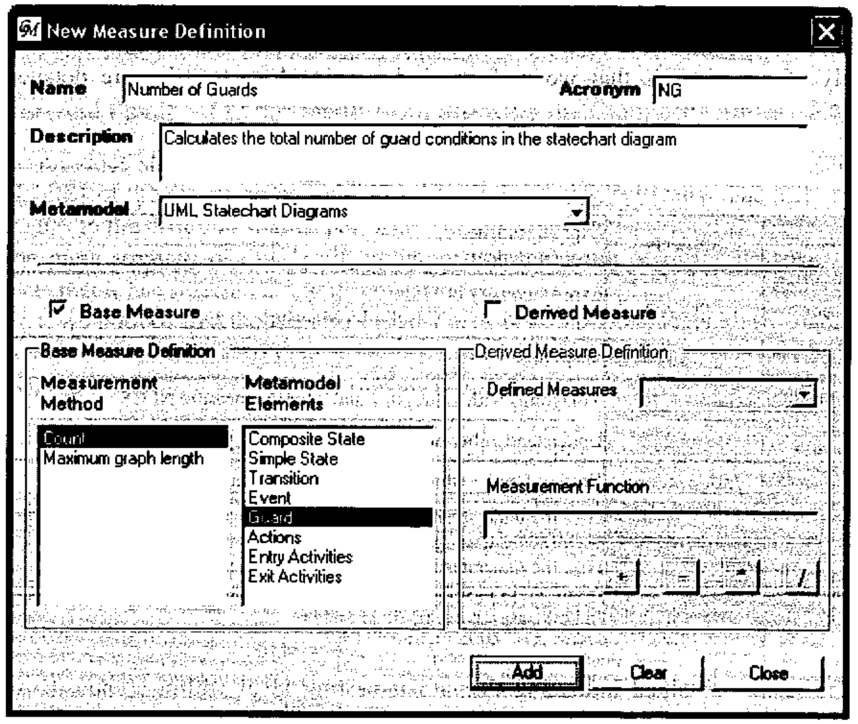
More details about this third family of experiments can be found in [10].

5 GenMETRIC

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

The two key characteristics of GenMETRIC are:

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



Name: Number of Guards
Acronym: NG
Description: Calculates the total number of guard conditions in the statechart diagram
Metamodel: UML Statechart Diagrams
☑ Base Measure
☐ Derived Measure
Base Measure Definition
Measurement Method: Count, Maximum graph length
Metamodel Elements: Composite State, Simple State, Transition, Event, Guard, Actions, Entry Activities, Exit Activities
Derived Measure Definition
Defined Measures
Measurement Function

Fig. 5. New measure definition in GenMETRIC

6 Conclusions

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

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

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

As future work, we are aware that it is necessary to continue refining the proposed measures and even to define some new ones (if necessary). This way we could have an adequate and useful set of measures for measuring the quality characteristics of UML statechart diagrams. Further empirical studies must also be preformed in order to reach a definitive and strong empirical validation of all the measures.

We will also try to extend the definition of measures to some other quality characteristics, such as modifiability, that also affect the maintainability of the diagrams.

Acknowledgements

This research is part of the MECENAS and the IDONEO projects (PBI06-0024, PAC08-0160-6141) financed by "Consejería de Ciencia y Tecnología de la Junta de Comunidades de Castilla-La Mancha" and the ESFINGE project (TIN2006-15175-C05-05) (TIN20005-24055-E) supported by the "Ministerio de Educación y Ciencia" (Spain).

References

1. Baroni, A.L., Braz, S., Brito e Abreu, F.: Using OCL to Formalize Object-Oriented Design Metrics Definitions. In: Proceedings of 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2002), Malaga, Spain, pp. 99–106 (2002)
2. Briand, L., El-Emam, K., Morasca, S.: Theoretical and Empirical Validation of Software Product Measures, ISERN (1995)

3. Briand, L., Morasca, S., Basili, V.: Property-Based Software Engineering Measurement. IEEE Transactions on Software Engineering 22(1), 68–86 (1996)
4. Briand, L., Morasca, S., Basili, V.: Response to: Comments on Property-Based Software Engineering Measurement: Refining the Additivity Properties. IEEE Transactions on Software Engineering 23, 196–197 (1997)
5. Briand, L., Wüst, J., Daly, J., Porter, V.: Exploring the Relationships between Design Measures and Software Quality in Object-Oriented Systems. The Journal of Systems and Software 51, 245–273 (2000)
6. Briand, L., Wüst, J., Lounis, H.: Investigating Quality Factors in Object-oriented Designs: An Industrial Case Study, Technical Report ISERN 98-29, version 2 (1998)
7. Calero, C., Piattini, M., Genero, M.: Method for Obtaining Correct Metrics. In: Proceedings of 3rd International Conference on Enterprise and Information Systems (ICEIS 2001), Setúbal, Portugal, pp. 779–784 (2001)
8. Cartwright, M.: An Empirical View of Inheritance. Information and Software Technology 40(4), 795–799 (1998)
9. Clavel, M., Durán, F., Eker, S., Lincoln, P., Marínez-Oliet, N., Meseguer, J., Talcott, C.: Maude 2.1.1 Manual, University of Illinois at Urbana-Champaign (2003)
10. Cruz-Lemus, J.A., Genero, M., Piattini, M.: Investigating the Nesting Level of Composite States in UML Statechart Diagrams. In: Proceedings of 9th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE 2005), Glasgow, United Kingdom, pp. 97–108 (2005)
11. Cruz-Lemus, J.A., Genero, M., Piattini, M., Morasca, S.: Improving the Experimentation for Evaluating the Effect of Composite States on the Understandability of UML Statechart Diagrams. In: Proceedings of 5th ACM-IEEE International Symposium on Empirical Software Engineering (ISESE 2006), Rio de Janeiro, Brazil, pp. 9–11 (2006)
12. Cruz-Lemus, J.A., Genero, M., Piattini, M., Toval Álvarez, J.A.: An Empirical Study of the Nesting Level of Composite States within UML Statechart Diagrams. In: Akoka, J., Liddle, S.W., Song, I.-Y., Bertolotto, M., Comyn-Wattiau, I., van den Heuvel, W.-J., Kolp, M., Trujillo, J., Kop, C., Mayr, H.C. (eds.) ER Workshops 2005. LNCS, vol. 3770, pp. 12–22. Springer, Heidelberg (2005)
13. Cruz-Lemus, J.A., Maes, A., Genero, M., Poels, G., Piattini, M.: The Impact of Structural Complexity on the Understandability of UML Statechart Diagrams, University of Ghent (2007)
14. Daly, J., Brooks, A., Miller, J., Roper, M., Wood, M.: An Empirical Study Evaluating Depth of Inheritance on Maintainability of Object-Oriented Software. Empirical Software Engineering 1(2), 109–132 (1996)
15. García, F., Serrano, M., Cruz-Lemus, J.A., Ruiz, F., Piattini, M.: Managing Software Process Measurement: a Metamodel-based Approach. Information Sciences 177, 2570–2586 (2007)
16. Mayer, R.E.: Multimedia Learning, Cambridge University Press (2001)
17. OMG, UML 2.0 OCL Final Adopted Specification, Object Management Group (2005)
18. Poels, G., Dedene, G.: Distance: A Framework for Software Measure Construction. Department of Applied Economics, Catholic University of Leuven, Belgium (1999)
19. Poels, G., Dedene, G.: Evaluating the Effect of Inheritance on the Modifiability of Object-Oriented Business Domain Models. In: Proceedings of 5th European Conference on Software Maintenance and Reengineering (CSMR 2001), Lisbon, Portugal, pp. 20–29 (2001)
20. Prechelt, L., Unger, B., Philippsen, M., Tichy, W.: A Controlled Experiment on Inheritance Depth as a Cost Factor for Code Maintenance. The Journal of Systems and Software 65, 115–126 (2003)
21. Webb, K.: Xholon Digital Watch Project (2006)

Printing: Mercedes-Druck, Berlin
Binding: Stein+Lehmann, Berlin

Lecture Notes in Computer Science

Sublibrary 2: Programming and Software Engineering

For information about Vols. 1–4554
please contact your bookseller or Springer

Vol. 5153: A. Rausch, R. Reussner, R. Mirandola, F. Plášil (Eds.), The Common Component Modeling Example. VIII, 460 pages. 2008.

Vol. 5142: J. Vitek (Ed.), ECOOP 2008 – Object-Oriented Programming. XIII, 694 pages. 2008.

Vol. 5140: J. Meseguer, G. Roşu (Eds.), Algebraic Methodology and Software Technology. XIII, 432 pages. 2008.

Vol. 5136: T.C.N. Graham, P. Palanque (Eds.), Interactive Systems
Design, Specification, and Verification. IX, 311 pages. 2008.

Vol. 5135: R. de Lemos, F. Di Giandomenico, C. Gacek, H. Muccini, M. Vieira (Eds.), Architecting Dependable Systems V. XIV, 343 pages. 2008.

Vol. 5119: S. Kounev, I. Gorton, K. Sachs (Eds.), Performance Evaluation: Metrics, Models and Benchmarks. X, 323 pages. 2008.

Vol. 5111: Q. Chen, C. Zhang, S. Zhang, Secure Transaction Protocol Analysis. XI, 234 pages. 2008.

Vol. 5095: I. Schieferdecker, A. Hartman (Eds.), Model Driven Architecture – Foundations and Applications. XIII, 446 pages. 2008.

Vol. 5091: B.P. Woolf, E. Aïmeur, R. Nkambou, S. Lajoie (Eds.), Intelligent Tutoring Systems. XXI, 832 pages. 2008.

Vol. 5089: A. Jedlitschka, O. Salo (Eds.), Product-Focused Software Process Improvement. XIV, 448 pages. 2008.

Vol. 5082: B. Meyer, J.R. Nawrocki, B. Meyer (Eds.), Balancing Agility and Formalism in Software Engineering. XI, 305 pages. 2008.

Vol. 5079: M. Alpuente, G. Vidal (Eds.), Static Analysis. X, 379 pages. 2008.

Vol. 5063: A. Vallecillo, J. Gray, A. Pierantonio (Eds.), Theory and Practice of Model Transformations. XII, 261 pages. 2008.

Vol. 5060: C. Rong, M.G. Jaatun, F.E. Sandnes, L.T. Yang, J. Ma (Eds.), Autonomic and Trusted Computing. XV, 666 pages. 2008.

Vol. 5055: K. Al-Begain, A. Heindl, M. Telek (Eds.), Analytical and Stochastic Modeling Techniques and Applications. XI, 323 pages. 2008.

Vol. 5052: D. Lea, G. Zavattaro (Eds.), Coordination Models and Languages. X, 347 pages. 2008.

Vol. 5051: G. Barthe, F.S. de Boer (Eds.), Formal Methods for Open Object-Based Distributed Systems. X, 259 pages. 2008.

Vol. 5048: K. Suzuki, T. Higashino, K. Yasumoto, K. El-Fakih (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2008. XII, 341 pages. 2008.

Vol. 5047: K. Suzuki, T. Higashino, A. Ulrich, T. Hasegawa (Eds.), Testing of Software and Communicating Systems. XII, 303 pages. 2008.

Vol. 5030: H. Mei (Ed.), High Confidence Software Reuse in Large Systems. XII, 388 pages. 2008.

Vol. 5026: F. Kordon, T. Vardanega (Eds.), Reliable Software Technologies – Ada-Europe 2008. XIV, 283 pages. 2008.

Vol. 5025: B. Paech, C. Rolland (Eds.), Requirements Engineering: Foundation for Software Quality. X, 205 pages. 2008.

Vol. 5020: J. Barnes, Ada 2005 Rationale. IX, 267 pages. 2008.

Vol. 5016: M. Bernardo, P. Degano, G. Zavattaro (Eds.), Formal Methods for Computational Systems Biology. X, 538 pages. 2008.

Vol. 5014: J. Cuellar, T. Maibaum, K. Sere (Eds.), FM 2008: Formal Methods. XIII, 436 pages. 2008.

Vol. 5007: Q. Wang, D. Pfahl, D.M. Raffo (Eds.), Making Globally Distributed Software Development a Success Story. XIV, 422 pages. 2008.

Vol. 5002: H. Giese (Ed.), Models in Software Engineering. X, 322 pages. 2008.

Vol. 4989: J. Garrigue, M.V. Hermenegildo (Eds.), Functional and Logic Programming. XI, 337 pages. 2008.

Vol. 4970: M. Nagl, W. Marquardt (Eds.), Collaborative and Distributed Chemical Engineering. XII, 851 pages. 2008.

Vol. 4966: B. Beckert, R. Hähnle (Eds.), Tests and Proofs. X, 193 pages. 2008.

Vol. 4954: C. Pautasso, É. Tanter (Eds.), Software Composition. X, 263 pages. 2008.

Vol. 4951: M. Luck, L. Padgham (Eds.), Agent-Oriented Software Engineering VIII. XIV, 225 pages. 2008.

Vol. 4949: R.M. Hierons, J.P. Bowen, M. Harman (Eds.), Formal Methods and Testing. XIII, 367 pages. 2008.

Vol. 4937: M. Dumas, R. Heckel (Eds.), Web Services and Formal Methods. IX, 169 pages. 2008.

Vol. 4922: M. Broy, I.H. Krüger, M. Meisinger (Eds.), Model-Driven Development of Reliable Automotive Services. XVIII, 183 pages. 2008.

Vol. 4916: S. Leue, P. Merino (Eds.), Formal Methods for Industrial Critical Systems. X, 251 pages. 2008.

Vol. 4909: I. Eusgeld, F.C. Freiling, R. Reussner (Eds.), Dependability Metrics. XI, 305 pages. 2008.

Vol. 4906: M. Cebulla (Ed.), Object-Oriented Technology. VIII, 204 pages. 2008.

Vol. 4902: P. Hudak, D.S. Warren (Eds.), Practical Aspects of Declarative Languages. X, 333 pages. 2007.

Vol. 4899: K. Yorav (Ed.), Hardware and Software: Verification and Testing. XII, 267 pages. 2008.

Vol. 4895: J.J. Cuadrado-Gallego, R. Braungarten, R.R. Dumke, A. Abran (Eds.), Software Process and Product Measurement. X, 203 pages. 2008.

Vol. 4888: F. Kordon, O. Sokolsky (Eds.), Composition of Embedded Systems. XII, 221 pages. 2007.

Vol. 4880: S. Overhage, C.A. Szyperski, R. Reussner, J.A. Stafford (Eds.), Software Architectures, Components, and Applications. X, 249 pages. 2008.

Vol. 4849: M. Winckler, H. Johnson, P. Palanque (Eds.), Task Models and Diagrams for User Interface Design. XIII, 299 pages. 2007.

Vol. 4839: O. Sokolsky, S. Taşıran (Eds.), Runtime Verification. VI, 215 pages. 2007.

Vol. 4834: R. Cerqueira, R.H. Campbell (Eds.), Middleware 2007. XIII, 451 pages. 2007.

Vol. 4829: M. Lumpe, W. Vanderperren (Eds.), Software Composition. VIII, 281 pages. 2007.

Vol. 4824: A. Paschke, Y. Biletskiy (Eds.), Advances in Rule Interchange and Applications. XIII, 243 pages. 2007.

Vol. 4821: J. Bennedsen, M.E. Caspersen, M. Kölling (Eds.), Reflections on the Teaching of Programming. X, 261 pages. 2008.

Vol. 4807: Z. Shao (Ed.), Programming Languages and Systems. XI, 431 pages. 2007.

Vol. 4799: A. Holzinger (Ed.), HCI and Usability for Medicine and Health Care. XVI, 458 pages. 2007.

Vol. 4789: M. Butler, M.G. Hinchey, M.M. Larrondo-Petrie (Eds.), Formal Methods and Software Engineering. VIII, 387 pages. 2007.

Vol. 4767: F. Arbab, M. Sirjani (Eds.), International Symposium on Fundamentals of Software Engineering. XIII, 450 pages. 2007.

Vol. 4765: A. Moreira, J. Grundy (Eds.), Early Aspects: Current Challenges and Future Directions. X, 199 pages. 2007.

Vol. 4764: P. Abrahamsson, N. Baddoo, T. Margaria, R. Messnarz (Eds.), Software Process Improvement. XI, 225 pages. 2007.

Vol. 4762: K.S. Namjoshi, T. Yoneda, T. Higashino, Y. Okamura (Eds.), Automated Technology for Verification and Analysis. XIV, 566 pages. 2007.

Vol. 4758: F. Oquendo (Ed.), Software Architecture. XVI, 340 pages. 2007.

Vol. 4757: F. Cappello, T. Herault, J. Dongarra (Eds.), Recent Advances in Parallel Virtual Machine and Message Passing Interface. XVI, 396 pages. 2007.

Vol. 4753: E. Duval, R. Klamma, M. Wolpers (Eds.), Creating New Learning Experiences on a Global Scale. XII, 518 pages. 2007.

Vol. 4749: B.J. Krämer, K.-J. Lin, P. Narasimhan (Eds.), Service-Oriented Computing – ICSOC 2007. XIX, 629 pages. 2007.

Vol. 4748: K. Wolter (Ed.), Formal Methods and Stochastic Models for Performance Evaluation. X, 301 pages. 2007.

Vol. 4741: C. Bessière (Ed.), Principles and Practice of Constraint Programming – CP 2007. XV, 890 pages. 2007.

Vol. 4735: G. Engels, B. Opdyke, D.C. Schmidt, F. Weil (Eds.), Model Driven Engineering Languages and Systems. XV, 698 pages. 2007.

Vol. 4716: B. Meyer, M. Joseph (Eds.), Software Engineering Approaches for Offshore and Outsourced Development. X, 201 pages. 2007.

Vol. 4709: F.S. de Boer, M.M. Bonsangue, S. Graf, W.-P. de Roever (Eds.), Formal Methods for Components and Objects. VIII, 297 pages. 2007.

Vol. 4680: F. Saglietti, N. Oster (Eds.), Computer Safety, Reliability, and Security. XV, 548 pages. 2007.

Vol. 4670: V. Dahl, I. Niemelä (Eds.), Logic Programming. XII, 470 pages. 2007.

Vol. 4652: D. Georgakopoulos, N. Ritter, B. Benatallah, C. Zirpins, G. Feuerlicht, M. Schoenherr, H.R. Motahari-Nezhad (Eds.), Service-Oriented Computing ICSOC 2006. XVI, 201 pages. 2007.

Vol. 4640: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development IV. IX, 191 pages. 2007.

Vol. 4634: H. Riis Nielson, G. Filé (Eds.), Static Analysis. XI, 469 pages. 2007.

Vol. 4620: A. Rashid, M. Aksit (Eds.), Transactions on Aspect-Oriented Software Development III. IX, 201 pages. 2007.

Vol. 4615: R. de Lemos, C. Gacek, A. Romanovsky (Eds.), Architecting Dependable Systems IV. XIV, 435 pages. 2007.

Vol. 4610: B. Xiao, L.T. Yang, J. Ma, C. Muller-Schloer, Y. Hua (Eds.), Autonomic and Trusted Computing. XVIII, 571 pages. 2007.

Vol. 4609: E. Ernst (Ed.), ECOOP 2007 – Object-Oriented Programming. XIII, 625 pages. 2007.

Vol. 4608: H.W. Schmidt, I. Crnković, G.T. Heineman, J.A. Stafford (Eds.), Component-Based Software Engineering. XII, 283 pages. 2007.

Vol. 4591: J. Davies, J. Gibbons (Eds.), Integrated Formal Methods. IX, 660 pages. 2007.

Vol. 4589: J. Münch, P. Abrahamsson (Eds.), Product-Focused Software Process Improvement. XII, 414 pages. 2007.

Vol. 4574: J. Derrick, J. Vain (Eds.), Formal Techniques for Networked and Distributed Systems – FORTE 2007. XI, 375 pages. 2007.

Vol. 4556: C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction, Part III. XXII, 1020 pages. 2007.

Vol. 4555: C. Stephanidis (Ed.), Universal Access in Human-Computer Interaction, Part II. XXII, 1066 pages. 2007.