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, 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 on the series home page at *http://www.springer.de/comp/lncs/*

Proposals for publication should be sent to the LNCS Editorial, Tiergartenstr. 17, 69121 Heidelberg, Germany

E-mail: lncs@springer.de

ISSN 0302-9743

ISBN 3-540-42215-3

9 783540 422150

Lecture Notes in Computer Science

Lecture Notes in Artificial Intelligence

http://www.springer.de

Dittrich · Geppert
Norrie (Eds.)

LNCS 2068

Advanced Information Systems Engineering

CAiSE 2001

Andreas Geppert
Maria C. Norrie (Eds.)

Advanced Information Systems Engineering

13th International Conference, CAiSE 2001
Interlaken, Switzerland, June 2001
Proceedings

Springer

Klaus R. Dittrich Andreas Geppert
Moira C. Norrie (Eds.)

Advanced Information Systems Engineering

13th International Conference, CAiSE 2001
Interlaken, Switzerland, June 4-8, 2001
Proceedings

Springer
Berlin
Heidelberg
New York
Barcelona
Hong Kong
London
Milan
Paris
Singapore
Tokyo

Springer

CAiSE 2001 Organization

Advisory Committee

Janis Bubenko Jr.
Kungl. Tekniska Högskolan,
Stockholm,
Sweden

Arne Sölvberg
The Norwegian University of Science
and Technology, Trondheim,
Norway

General Chair

Moira Norrie
ETH Zurich, Switzerland

Program Chair

Klaus R. Dittrich
University of Zurich, Switzerland

Panel and Tutorial Chair

Michel Leonard
University of Geneva,
Switzerland

Workshop and Poster Chair

Stefano Spaccapietra
EPFL Lausanne,
Switzerland

Organizing Committee

Carl A. Zehnder
Andrea Lombardoni
Beat Signer
ETH Zurich,
Switzerland

Program Committee

Witold Abramowicz	University of Poznan, Poland
Hans-Jürgen Appelrath	University of Oldenburg, Germany
Alex Borgida	Rutgers University, USA
Sjaak Brinkkemper	Baan Company R&D, The Netherlands
Janis Bubenko	Royal Institute of Technology and Stockholm University, Sweden
Silvana Castana	University of Milan, Italy
Panos Constantopoulos	University of Crete, Greece
Jan Dietz	Delft University of Technology, The Netherlands
Klaus Dittrich (Chair)	University of Zurich, Switzerland
Mariagrazia Fugini	Politecnico di Milano, Italy
Antonio Furtado	PUC, Rio de Janeiro, Brazil
Andreas Geppert	University of Zurich, Switzerland
Martin Glinz	University of Zurich, Switzerland
Jean-Luc Hainaut	University of Namur, Belgium
Juhani Iivari	University of Oulu, Finland
Stefan Jablonski	University of Erlangen, Germany
Christian S. Jensen	University of Aalborg, Denmark
Manfred Jeusfeld	Tilburg University, The Netherlands
Paul Johannesson	Stockholm University and Royal Institute of Technology, Sweden
Gerti Kappel	Johannes Kepler University of Linz, Austria
Pericles Loucopoulos	UMIST, United Kingdom
Kalle Lyytinen	University of Jyväskylä, Finland
Neil Maiden	City University, England
Michele Missikoff	IASI-CNR, Italy
John Mylopoulos	University of Toronto, Canada
Oscar Nierstrasz	University of Berne, Switzerland
Andreas Oberweis	University of Frankfurt, Germany
Antoni Olivè	Polytechnical University of Catalunya, Spain
Andreas Opdahl	University of Bergen, Norway
Norman Paton	University of Manchester, United Kingdom
Barbara Pernici	Politecnico di Milano, Italy
Alain Pirotte	University of Louvain, Belgium
Klaus Pohl	University of Essen, Germany
Colette Rolland	University of Paris 1, France
Michel Scholl	INRIA, France
Michael Schrefl	Johannes Kepler University of Linz, Austria
Amilcar Sernadas	Technical University of Lisboa, Portugal
Arne Sølvberg	Norwegian Institute of Technology, Norway
Martin Staudt	Swiss Life, Switzerland
Rudi Studer	University of Karlsruhe, Germany
Costantinos Thanos	CNR-IEI, Italy

Workflow Management

The P2P Approach to Interorganizational Workflows 140
Wil M.P. van der Aalst, Mathias Weske
(Eindhoven University of Technology, The Netherlands)

Relaxed Soundness of Business Processes 157
Juliane Dehnert (Technical University Berlin, Germany), Peter Rittgen
(University of Koblenz-Landau, Germany)

Developing E-Services for Composing E-Services 171
Fabio Casati, Mehmet Sayal, Ming-Chien Shan
(Hewlett-Packard Laboratories, USA)

Data Models and Design

Data Models with Multiple Temporal Dimensions: Completing the Picture 187
Carlo Combi, Angelo Montanari (University of Udine, Italy)

Querying Data-Intensive Programs for Data Design 203
Jianhua Shao, Xingkun Liu, G. Fu, Suzanne M. Embury, W.A. Gray
(Cardiff University, Wales)

Consistency Management of Financial XML Documents 219
Andrea Zisman (City University, England),
Adamantia Athanasopoulou (Singular International SA, Greece)

Reuse and Method Engineering

The Relationship between Organisational Culture and the Deployment of Systems Development Methodologies 234
Juhani Iivari (University of Oulu, Finland),
Magda Huisman (Potchefstroom University, South Africa)

Increasing Reusability in Information Systems Development by Applying Generic Methods 251
Silke Eckstein, Peter Ahlbrecht, Karl Neumann
(TU Braunschweig, Germany)

An Assembly Process Model for Method Engineering 267
Jolita Ralyté, Colette Rolland (Université Paris Sorbonne, France)

Process Reuse Architecture 284
Soeli T. Fiorini, Julio C. Sampaio do Prado Leite,
Carlos J. Pereira de Lucena
(Pontifícia Universidade Católica do Rio de Janeiro, Brazil)

XML and Information System Integration

Using a Metadata Software Layer in Information Systems Integration 299
Mark Roantree (Dublin City, Ireland),
Jessie B. Kennedy, Peter J. Barclay (Napier University, Scotland)

Distributed Information Search with Adaptive Meta-Search Engines 315
Lieming Huang, Ulrich Thiel, Matthias Hemmje, Erich J. Neuhold
(GMD-IPSI, Germany)

A Semantic Approach to Integrating XML and Structured Data Sources .. 330
Peter McBrien (Imperial College, England),
Alexandra Poulovassilis (University of London, England)

XML-Based Integration of GIS and Heterogeneous Tourism Information .. 346
Franz Pühretmair, Wolfram Wöß
(Johannes Kepler University Linz, Austria)

Evolution

Objects Control for Software Configuration Management 359
Jacky Estublier (Dassault Systèmes & LSR, Grenoble University, France)

Coordination Technologies for Managing Information System Evolution ... 374
Luis Filipe Andrade (ATX Software S.A., Portugal),
José Luiz Fiadeiro (University of Lisbon, Portugal)

Using Metrics to Predict OO Information Systems Maintainability 388
Marcela Genero, José Olivas, Mario Piattini, Francisco Romero
(University of Castilla-La Mancha, Spain)

Conceptual Modelling

A Rigorous Metamodel for UML Static Conceptual Modelling of Information Systems 402
Régine Laleau (CNAM, France),
Fiona Polack (University of York, England)

Taxonomies and Derivation Rules in Conceptual Modelling 417
Antoni Olivé (Universitat Politècnica de Catalunya, Spain)

Using UML Action Semantics for Executable Modeling and Beyond 433
Gerson Sunyé, François Pennaneac'h, Wai-Ming Ho,
Alain Le Guennec, Jean-Marc Jézéquel (IRISA, France)

Design and Implementation of a UML-Based Design Repository 448
Rudolf K. Keller, Jean-François Bédard, Guy Saint-Denis
(Université de Montréal, Canada)

Using Metrics to Predict OO Information Systems Maintainability

Marcela Genero, José Olivas, Mario Piattini, and Francisco Romero

Department of Computer Science
University of Castilla-La Mancha
Ronda de Calatrava, 5
13071, Ciudad Real, Spain
{mgenero, jaolivas, mpiattin, fpromero}@inf-cr.uclm.es

Abstract. The quality of object oriented information systems (OOIS) depends greatly on the decisions taken at early phases of their development. As an early available artifact the quality of the class diagram is crucial to the success of system development. Class diagrams lay the foundation for all later design work. So, their quality heavily affects the product that will be ultimately implemented. Even though the appearance of the Unified Modeling Language (UML) as a standard of modelling OOIS has contributed greatly towards building quality OOIS, it is not enough. Early availability of metrics is a key factor in the successful management of OOIS development. The aim of this paper is to present a set of metrics for measuring the structural complexity of UML class diagrams and to use them for predicting their maintainability that will heavily be correlated with OOIS maintainability.

Keywords. object oriented information systems maintainability, object oriented metrics, class diagrams complexity, UML, fuzzy deformable prototypes, prediction models

1 Introduction

A widely accepted principle in software engineering is that the quality of a software product should be assured in the early phases of its life cycle. In a typical OOIS design at the early phases, a class diagram is first built. The class diagram is not merely the basis of modelling the persistent system data. In OO modelling, where data and process are closely linked, class diagrams provide the solid foundation for the design and implementation of OOIS.

As an early available, key analysis artifact the quality of the class diagram is crtucial to the success of system development. Generally, problems in the artifacts produced in the initial phases of system development propagate to the artifacts produced in later stages, where they are much more costly to identify and correct [2]. As a result, improving the quality of class diagrams, will therefore be a major step towards the quality improvement of the OOIS development. The appearance of UML [20], as standard OO modelling language, should contribute to this. Despite this, we have to be aware that a standard modelling language can only give us syntax and semantics to work with, but it cannot tell us whether a "good" model has been

produced. Naturally, even when language is mastered, there is no guarantee that the models produced will be good. Therefore, it is necessary to assess their quality.

The definition of the different characteristics that compose the concept of "quality" is not enough on its own in order to ensure quality in practice, as people will generally make different interpretations of the same concept. Software measurement plays an important role in this sense because metrics provide a valuable and objective insight into specific ways of enhancing each of the software quality characteristics. Measurement data can be gathered and analysed to assess current product quality, to predict future quality, and to drive quality improvement initiatives [27].

Quality is a multidimensional concept, composed of different characteristics such as functionality, reliability, usability, efficiency, maintainability and portability [15]. This paper focuses on UML class diagram maintainability, because maintainability has been and continues to be one of the pressing challenges facing any software development department. For our purpose we distinguish the following maintainability sub-characteristics:

- UNDERSTANDABILITY. The ease with which the class diagram can be understood.
- ANALYSABILITY. The capability of the class diagram to be diagnosed for deficiencies or to identify parts to be modified.
- MODIFIABILITY. The capability of the class diagram to enable a specified modification to be implemented.

But these maintainability sub-characteristics are external quality attributes that can only be measured late in the OOIS life cycle. Therefore it is necessary to find early indicators of such qualities based, for example, on the structural properties of class diagrams [4].

The availability of significant measures in the early phases of the software development life-cycle allows for better management of the later phases, and more effective quality assessment when quality can be more easily affected by corrective actions [3]. They allow IS designers:

1. a quantitative comparison of design alternatives, and therefore and objective selection among several class diagram alternatives with equivalent semantic content.
2. a prediction of external quality characteristics, like maintainability in the initial phases of the IS life cycle and a better resource allocation based on these predictions.

After performing a thorough review of several OO metric proposals [9],[18],[7],19], specially focusing in those that can be applied to class diagrams at a high level design stage we have proposed new ones [14] related to the structural complexity of class diagrams due to the usage of relationships (associations, dependencies, generalisations, aggregations). But proposing metrics it is not enough to assure that they really are fruitful in practice. Empirical validation is a crucial task for the success of software measurement [17],[13],[26],[1].

Our main motivation is to present metrics [14] for measuring UML class diagram structural complexity (internal quality attribute) and secondly demonstrate through experimentation that it can be used to predict UML class diagram maintainability (external quality attribute), which will strongly influence OOIS maintainability.

This paper is organised in the following way: In section 2 we will present a set of metrics for measuring UML class diagram structural complexity. In section 3 we

We allowed one week to do the experiment, i.e., each subject had carry out the test alone, and could use unlimited time to solve it.

After completion of the tasks subjects were asked to complete a debriefing questionnaire. This questionnaire included (i) personal details and experience, (ii) opinions on the influence of different components of UML Diagrams, such as: classes, attributes, associations, generalisations, etc... on their maintainability.

3.3 Experimental Design and Data Collection

The INDEPENDENT VARIABLES are those metrics proposed in sections 2.1 and 2.2.

The DEPENDENT VARIABLES are three of the maintainability sub-characteristics: understandability, analysability and modifiability measured according to subject's rating.

We decided to give our subjects as much time as they needed to finish the test they had to carry out. All tests were considered valid because all of the subjects have at least medium experience in building UML class diagrams and developing OOIS (this fact was corroborated analysing the responses of the debriefing questionnaire).

3.4 Construction of Fuzzy Deformable Prototypes to Characterise UML Class Diagram Maintainability

We have used an extension of the traditional Knowledge Discovery in Databases (KDD) [12]: the Fuzzy Prototypical Knowledge Discovery (FPKD) that consists of the search for fuzzy prototypes [28] that characterise the maintainability of an UML class diagram.

In the rest of this section we will explain each of the steps we have followed in the FPKD (see figure 1).

Selection of the Target Data. We have taken as a start set a relational database that contains 476 records (with 16 fields,13 represent metrics values, 3 represent maintainability sub-characteristics) obtained from the calculation of the metric values (for each class diagram) and the responses of the experiment given by the subjects.

Preprocessing. The Data-Cleaning was not necessary because we didn't find any errors.

Transformation. This step was performed doing different tasks:

- SUMMARISING SUBJECT RESPONSES. We built a unique table with 28 records (one record for each class diagram) and 17 fields (13 metrics and 3 maintainability sub-characteristics). This table is shown in Appendix A). The metric values were calculated measuring each diagram, and the values for each maintainability sub-characteristics were obtained aggregating subjects's rating using the mean of them.

- CLUSTERING BY REPERTORY GRIDS. In order to detect the relationships between the class diagrams, for obtaining those which are easy, medium or difficult to maintain (based on subject rates of each maintainability sub-characteristics), we have carried out a hierarchical clustering process by Repertory Grids. The set of elements is constituted by the 28 class diagrams, the constructions are the intervals of values of the subjects'rating. To accomplish an analysis of clusters on elements, we have built a proximity matrix that represents the different similarities of the elements, a matrix of 28 x 28 elements (the diagrams) that above the diagonal represents the distances between the different cycles. Converting these values to percentages, a new table is created and the application of Repertory Grids Analysis Algorithm returns a graphic as a final result (see figure 1).

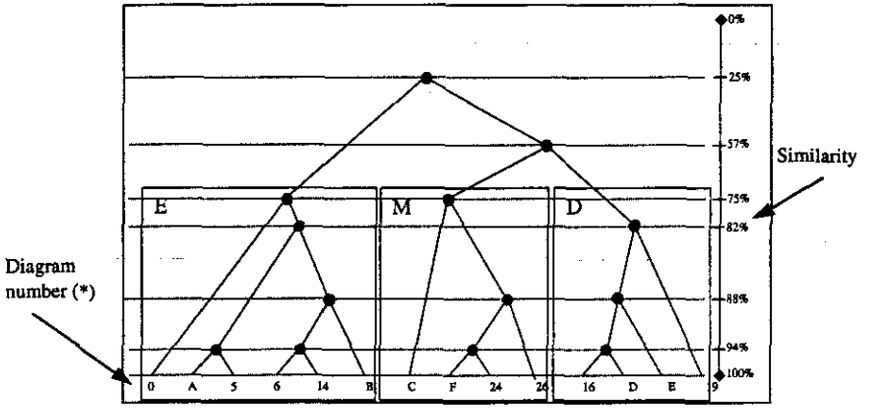

Fig. 1. Clustering results (E: Easy to maintain, M: Medium to maintain, D: Difficult to maintain)

(*) We have grouped some class diagrams assigning them one letter because they have 100% of similarity (see appendix A)

- DATA MINING. The selected algorithm for data mining process was summarise functions. Table 1 shows the parametric definition of the prototypes. These parameters will be modified taking into account the degree of affinity of a new class diagram with the prototypes. With the new modified prototype we will be able to predict the maintainability of a new class diagram.
- FORMAL REPRESENTATION OF CONCEPTUAL PROTOTYPES. The prototypes have been represented as fuzzy numbers, which are going to allow us to obtain a degree of membership in the concept. For the sake of simplicity in the model, they have been represented by triangular fuzzy numbers. Therefore, in order to construct the prototypes (triangular fuzzy numbers) we only need to know their centerpoints ("center of the prototype"), which are obtained by normalising and aggregating the metric values corresponding to the class diagrams of each of the prototypes (see figure 2).

same degree of experience, and if the subjects have more experience minor inaccuracies could be introduced by subjects.
- LEARNING EFFECTS. All the tests in each experiment were put in a different order, to avoid learning effects. Subjects were required and controlled to answer in the order in which test appeared.
- FATIGUE EFFECTS. On average the experiment lasted for less than one hour, so fatigue was not very relevant. Also, the different order in the tests helped to avoid these effects.
- PERSISTENCE EFFECTS. In order to avoid persistence effects, the experiment was run with subjects who had never done a similar experiment.
- SUBJECT MOTIVATION. All the professors who were involved in this experiment have participated voluntarily, in order to help us in our research. We motivated students to participate in the experiment, explaining to them that similar tasks to the experimental ones could be done in exams or practice by students, so they wanted to take the most of the experiment.
- OTHER FACTORS. Plagiarism and influence between students really could not be controlled. Students were told that taking with each other was forbidden, but they did the experiment alone without any control, so we had to trust them as far as that was concerned.

Seeing the results of the experiment we can conclude that empirical evidence of the existing relationship between the independent and the dependent variables exists. But only by replicating controlled experiments, where the measures would be varied in a controlled manner and all the other factors would be kept constant, could really demonstrate causality.

Threats to External Validity. The greater the external validity, the more the results of an empirical study can be generalised to actual software engineering practice. Two threat of validity have been identified which limit the ability to apply any such generalisation:

- MATERIALS AND TASKS USED. In the experiment we tried to use class diagrams and tasks which can be representative of real cases, but more empirical studies taking "real cases" from software companies must be done.
- SUBJECTS. To solve the difficulty of obtaining professional subjects, we used professors and advanced students from software engineering courses. We are aware that more experiments with practitioners and professionals must be carried out in order to be able to generalise these results. However, in this case, the tasks to be performed do not require high levels of industrial experience, so, experiments with students could be appropriate [1].

In general in order to extract a final conclusion we need to replicate this experiment with a greater number of subjects, including practitioners. After doing replication we will have a cumulative body of knowledge; which will lead us to confirm if the presented metrics could really be used as early quality indicators, and could be used to predict UML class diagrams maintainability.

4 Prediction of UML Class Diagram Maintainability

Using Fuzzy Deformable Prototypes [21],[22], we can deform the most similar prototype to a new class diagram, and define the factors for a new situation, using a linear combination with the degrees of membership as coefficients. We will show an example of how to deform the fuzzy prototypes found in section 3.5. Given the normalised values corresponding to a new class diagram:

NC	NA	NM	NAssocR	NAssocVC	NAggR	NAggRVC	NAggH	NDepR	NDepVC	NGenR	NGH	MaxDIT
0.7	05	0.69	0.71	0.48	0.67	0.45	0.67	0.75	0.43	0.83	1	0.4

The final average is 0.64. The affinity with the prototypes is shown in figure 3.

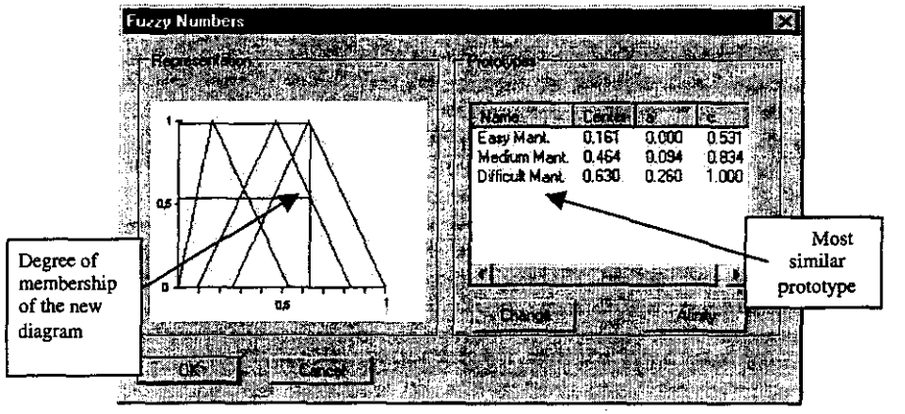

Fig. 3. Affinity of the real case with the prototypes

The most similar prototype for this new class diagram is "Difficult to maintain", with a degree of membership of 0.98. Then, the prediction is:

	Understandability	Analisability	Modifiability
Average	6	6	6
Maximum	6	6	7
Minimum	6	5	6

We want to highlight that this a first approach to predict UML class diagram maintainability, we need "real data" about UML class diagram maintainability efforts, like time spent in maintenance tasks in order to predict data that can be highly useful to software designers and developers.

2. Boehm, B. *Software Engineering Economics*. Prentice-Hall (1981).
3. Briand L., Morasca S. and Basili V. Defining and Validating Measures for Object-Based high-level design. *IEEE Transactions on Software Engineering*. Vol. 25 No. 5, (1999) 722-743.
4. Briand L., Arisholm S., Counsell F., Houdek F., and Thévenod-Fosse P. Empirical Studies of Object-Oriented Artifacts, Methods, and Processes: State of the Art and Future Directions. *Technical Report IESE 037.99/E, Fraunhofer Institute for Experimental Software Engineering*, Kaiserslautern, Germany, (1999).
5. Briand L., Bunse C. and Daly J. A Controlled Experiment for evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs. *Technical Report IESE 002.99/E, Fraunhofer Institute for Experimental Software Engineering*, Kaiserslautern, Germany, (1999).
6. Briand L., Wüst J., Daly J. and Porter D. Exploring the relationships between design measures and software quality in object-oriented systems. *The Journal of Systems and Software 51*, (2000) 245-273.
7. Brito e Abreu, F. and Carapaçua, R. (1994). Object-Oriented Software Engineering: Measuring and controllong the development process. *4th Int Conference on Software Quality*, Mc Lean, USA.
8. Brito e Abreu F., Zuse H., Sahraoui H. and Melo W. Quantitative Approaches in Object-Oriented Software Engineering. *Object-Oriented technology: ECOOP'99 Workshop Reader*, Lecture Notes in Computer Science 1743, Springer-Verlag, (1999) 326-337.
9. Chidamber S. and Kemerer C. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*. Vol. 20 No. 6, (1994) 476-493.
10. Daly J., Brooks A., Miller J., Roper M. and Wood M. Evaluating Inheritance Depth on the Maintainability of Object-Oriented Software. *Empirical Software Engineering*, 1, Kluwer Academic Publishers, Boston, (1996) 109-132.
11. Derr K. *Applying OMT*. SIGS Books, New York. (1995).
12. Fayyad U., Piatetsky-Shapiro G. and Smyth P. The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Communications of the ACM*, Vol. 39 No. 11, (1996) 27 – 34.
13. Fenton N. and Pfleeger S. *Software Metrics: A Rigorous Approach*. 2nd. edition. London, Chapman & Hall, (1997).
14. Genero, M., Piattini, M. and Calero, C. Early Measures For UML class diagrams. *L'Objet*. Hermes Science Publications, Vo.l 6 No. 4, (2000) 489-515.
15. ISO/IEC 9126-1.2. Information technology- Software product quality – Part 1: Quality model, (1999).
16. Henderson-Sellers B. *Object-Oriented Metrics - Measures of complexity*. Prentice-Hall, Upper Saddle River, New Jersey, (1996).
17. Kitchenham, B., Pflegger, S. and Fenton, N. Towards a Framework for Software Measurement Validation. *IEEE Transactions of Software Engineering*, Vol. 21 No. 12, (1995) 929-943.
18. Lorenz M. and Kidd J. *Object-Oriented Software Metrics: A Practical Guide*. Prentice Hall, Englewood Cliffs, New Jersey, (1994).
19. Marchesi M. OOA Metrics for the Unified Modeling Language. *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering*, (1998) 67-73.
20. Object Management Group. *UML Revision Task Force. OMG Unified Modeling Language Specification, v. 1.3. document ad/99-06-08*, (1999).
21. Olivas J. A. and Romero F. P. FPKD. Fuzzy Prototypical Knowledge Discovery. Application to Forest Fire Prediction. *Proceedings of the SEKE'2000*, Knowledge Systems Institute, Chicago, Ill. USA, (2000) 47 – 54.

22. Olivas J. A. Contribution to the Experimental Study of the Prediction based on Fuzzy Deformable Categories, PhD Thesis, University of Castilla-La Mancha, Spain, (2000).
23. Poels G. On the use of a Segmentally Additive Proximity Structure to Measure Object Class Life Cycle Complexity. *Software Measurement : Current Trends in Research and Practice*, Deutscher Universitäts Verlag, (1999), 61-79,.
24. Poels G. On the Measurement of Event-Based Object-Oriented Conceptual Models. *4th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, June 13, Cannes, France, (2000).
25. Poels, G. and Dedene, G.. Measures for Assessing Dynamic Complexity Aspects of Object-Oriented Conceptual Schemes. *In: Proceedings of the 19th International Conference on Conceptual Modeling (ER 2000)*, Salt Lake City, (2000), 499-512.
26. Schneidewind, N. Methodology For Validating Software Metrics. *IEEE Transactions of Software Engineering*, Vol. 18 No. 5, (1992) 410-422.
27. Tian J. Taxonomy and Selection of Quality Measurements and Models. *Proceedings of SEKE'99, The 11th International Conference on Software Engineering & Knowledge Engineering*, June 16-19, (1999) 71-75.
28. Zadeh, L. A.A note on prototype set theory and fuzzy sets. *Cognition* 12, (1982), 291–297.
29. H. Zuse. *A Framework of Software Measurement*. Berlin, Walter de Gruyter, (1998).