nadas

ajo
LMEN

13 de Junio de 2001

Organizado por:

Grupo MADEIRA (Metodologías y Arquitecturas para la Difusión de Información Electrónica por la Red)

Colaboradores:

Dpto. de Lenguajes y Sistemas Informáticos

Facultad de Informática

UNIVERSIDAD de SEVILLA

Presentación

El proyecto DOLMEN (Distributed Objects, Languages, Models and Environments) es fruto de la colaboración entre seis grupos de investigación en distintos pero complementarios aspectos de la ingeniería del software de varias universidades españolas (Valencia, Murcia, Sevilla, Granada, Valladolid y Castilla-La Mancha). DOLMEN es un proyecto coordinado que toma el relevo del proyecto MENHIR, que fue subvencionado por la Comisión Interministerial de Ciencia y Tecnología (TIC97-0593-C05), y que supuso en gran avance en las interrelaciones de los distintos grupos que lo componían, como se demuestra en el nacimiento de DOLMEN.

Como todo proyecto, DOLMEN no sería una realidad sino fuese por la dotación económica que permita la presentación de los resultados de los investigadores que lo componen en distintos foros (nacionales e internacionales), la realización de reuniones de coordinación entre los miembros de los distintos grupos, la adquisición de equipos y material que permita la renovación y puesta al día de los investigadores, etc. En este caso, dicha dotación económica viene de la mano del Ministerio de Ciencia y Tecnología y los fondos FEDER (TIC2000-1673-C06).

DOLMEN continúa con la misma ilusión con la que nació MENHIR, con el objetivo de intercambiar experiencias y puesta en común de investigaciones en el marco de la ingeniería de requisitos en particular y en la ingeniería del software en general, con especial énfasis en el desarrollo de herramientas que asistan en el desarrollo de productos software e incluso en la automatización de determinadas tareas. Con esta idea, se organizan las primeras Jornadas de Trabajo DOLMEN, que tendrán lugar en Sevilla durante los días 12 y 13 de junio de 2001. Anteriores jornadas de trabajo (cinco en total) se organizaron en el seno del proyecto MENHIR, y tuvieron lugar, en Sevilla (noviembre de 1997), Valencia (febrero de 1998), Murcia (noviembre de 1998), Sedano (mayo de 1999) y Granada (marzo de 2000).

En estas Jornadas se han recibido un total de 16 trabajos de los seis grupos participantes, que se han distribuidos en 4 áreas de investigación:

- Lenguajes de modelado y Metodologías
- Reutilización de software
- Evolución de software
- Interfaces de usuario

Presidente de las Jornadas

Miguel Toro

Comité Organizador

Presidente: Jesús Torres
Vocales: Juan M. Cordero
Mª José Escalona
Mariano González
Rafael Martínez
Manuel Mejías
Juan A. Ortega
Antonia Reina

Grupos participantes

- Depto. de Sistemas Información y Computación. Universidad Politécnica de Valencia
- Depto. de Lenguajes y Sistemas Informáticos. Universidad de Murcia
- Depto. de Lenguajes y Sistemas Informáticos. Universidad de Sevilla
- Depto. de Lenguajes y Sistemas Informáticos. Universidad de Granada
- Depto. de Informática. Universidad de Valladolid
- Depto. de Informática. Universidad de Castilla-La Mancha

Índice

Metrics for UML class diagrams

Marcela Genero, Mario Piattini, Coral Calero
Alarcos Research Group. Department of Computer Science
University of Castilla-La Mancha
Ronda de Calatrava, 5. 13071, Ciudad Real (Spain)
{mgenero, mpiattin, ccalero}@inf-cr.uclm.es

Abstract. Class diagrams constitute a key artifact in the development of object-oriented information systems (OOIS), their quality is crucial because it has a great impact on the quality of the OOIS which is ultimately implemented. For that reason, we have defined a set of measures for evaluating the structural complexity (an internal quality attribute) of class diagrams made using the Unified Modelling Languages (UML). These measures could be useful to predict class diagram external quality characteristics, such as maintainability, early in the OOIS life-cycle. In order to demonstrate that those metrics serve the purpose they were defined for, we have put them under empirical validation by means of controlled experiments. The explanation of the steps followed to define the metrics and their validation constitute the main objectives of this paper.

1. INTRODUCTION

A great effort has been made in the field of software measurement in order to achieve better quality OOIS (Henderson-Sellers, 1996; Melton, 1996, Zuse, 1998; Fenton and Pflegger, 1997), but most of them pursue the goal of evaluating -by means of quantitative measures- the quality of the final product, i.e. the code or the advanced design. We believe that in order to get better OOIS we should focus on measuring the quality characteristics of early artifacts, such as class diagrams, and based on those measurements thereby obtain early in the life-cycle a prediction model for OOIS quality characteristics (ISO, 1999), like for example maintainability.

As class diagrams constitute a key artifact in the development of OOIS, their quality is crucial because it has a great impact on the quality of the OOIS which is ultimately implemented.

In response to the great demand for measures for measuring quality characteristics of class diagrams, such as maintainability (ISO, 9126) and after a thorough review of some of the existing OO measures that can be applied at a high level design stage (Chidamber and Kemerer, 1994; Lorenz and Kidd, 1994, Brito e Abreu and Carapaçua, 1994; Marchesi, 1998) we have proposed a set of measures for UML class diagram structural complexity in Genero et al. (2000). As maintainability is an external quality characteristic that can be evaluated once a product is finished or nearly finished, we center our work on measuring an internal quality characteristic, the structural complexity of class diagrams. Our idea is to use those measures to predict class diagram maintainability early in the OOIS development.

We have defined those measures in a methodological way including three main tasks: metric definition, theoretical validation and empirical validation. Through empirical validation we can

demonstrate with real evidence that the measures we proposed serve the purpose they were defined for and that they are fruitful in practice.

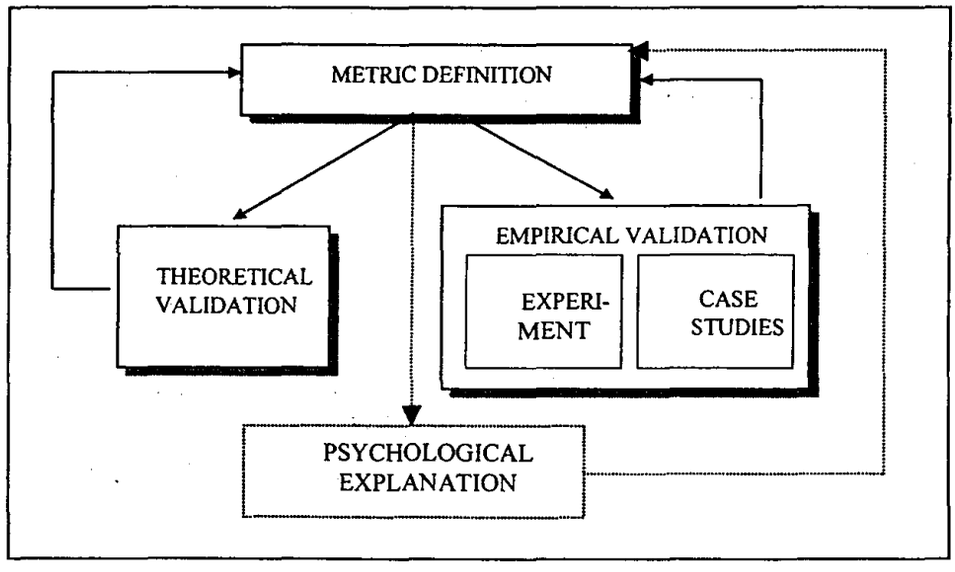
The objective of this paper is threefold:

1) To explain the method followed in the metric definition and validation (section 2)
2) To present metrics for UML class diagram structural complexity (see section 3)
3) To show a controlled experiment we have carried out in order to evaluate if there is empirical evidence that UML class diagram structural complexity metrics are correlated with maintainability sub-characteristics: such as understandability, analysability and modifiability (ISO, 1999) (see section 4).

Finally in section 5, we present some concluding remarks and future trends in metrics for object-oriented conceptual modelling using UML.

2. A FRAMEWORK FOR DEVELOPING AND VALIDATING METRICS

Metrics definition must be done in a methodological way, it is necessary to follow a number of steps to ensure the reliability of the proposed metrics. Figure 1 presents the method we apply for the metrics proposal (Calero et al., 2001).



In this figure we have four main activities:

- **Metrics definition.** The first step is the proposal of metrics. Although it looks simple, it is an important one in ensuring metrics are correctly defined. This definition is made taking into account the specific characteristics of the object models we want to measure and the experience of object-oriented designers.

- **Theoretical validation.** The second step is the formal validation of the metrics. The formal validation helps us to know when and how to apply the metrics. There are two main tendencies in metrics validation: the frameworks based on axiomatic approaches and the ones based on the measurement theory. The goal of the first ones is merely definitional. The most well-known frameworks of this type are those proposed by Weyuker (1988), Briand et al. (1996) and Morasca and Briand (1997). Since the goal of axiomatisation in software metrics research is primarily definitional, with the aim of providing a standard against which to validate software metrics, it is not so obvious that the risks outweigh the benefits (Kitchenham and Stell, 1997). The measurement theory-based frameworks (such as Zuse 1998 or Withmire, 1998) specify a general framework in which measures should be defined. The strength of measurement theory is the formulation of empirical conditions from which we can derive hypothesis of reality.

- **Empirical validation.** The goal of this step is to prove the practical utility of the proposed metrics. Although there are various ways of performing this step, basically we can divide the empirical validation into experimentation and case studies. Experimentation is usually made using controlled experiments and the case studies usually work with real data. Both of them are necessary, the controlled experiments for having a first approach and the case studies for making the results stronger. In both cases, the results are analyzed using either statistics tests or advanced data mining techniques.

- **Psychological explanation.** Ideally we will be able to explain the influence of the values of the metrics from a psychological point of view. Some authors, as Siau (1999), propose the use of cognitive psychology as a reference discipline in the engineering of methods and the studying of information modeling. In this sense, cognitive psychology theories such as the Adaptive Control of Thought (ACT, Anderson, 1983) could justify the influence of certain metrics in the database understandability. The knowledge of the limitation of human information processing capacity could also be helpful in establishing a threshold in the metrics for assuring the database quality.

As shown in figure 1, the process of defining and validating database metrics is evolutionary and iterative. As a result of the feedback, metrics could be redefined based on discarded theoretical, empirical or psychological validations.

3. DEFINITION OF METRICS FOR UML CLASS DIAGRAM STRUCTURAL COMPLEXITY

We only present here those metrics analyzed in Genero et al. (2000) which can be applied at class diagram level as a whole (see table 1). These metrics measure the structural complexity of UML class diagrams due to the use of relationships, such as associations, generalisations, aggregations and dependencies. We also consider traditional metrics such as, the number of classes, the number of attributes, etc.

Metric name	Metric definition
NUMBER OF CLASSES (NC)	The total number of classes.
NUMBER OF ATTRIBUTES (NA)	The total number of attributes.
NUMBER OF METHODS (NM)	The total number of methods
NUMBER OF ASSOCIATIONS (NAssoc)	The total number of associations
NUMBER OF AGGREGATION (NAgg)	The total number of aggregation relationships within a class diagram (each whole-part pair in an aggregation relationship)
NUMBER OF DEPENDENCIES (NDep)	The total number of dependency relationships
NUMBER OF GENERALISATIONS (NGen)	is defined as the total number of generalisation relationships within a class diagram (each parent-child pair in a generalisation relationship)
NUMBER OF GENERALISATIONS HIERARCHIES (NgenH)	The total number of generalisations hierarchies in a class diagram
MAXIMUM DIT	It is the maximum between the DIT value obtained for each class of the class diagram. The DIT value for a class within a generalisation hierarchy is the longest path from the class to the root of the hierarchy.
MAXIMUM HAGG	It is the maximum between the HAgg value obtained for each class of the class diagram. The HAgg value for a class within an aggregation hierarchy is the longest path from the class to the leaves.

Table 1. Metrics for UML class diagram structural complexity

4. EMPIRICAL VALIDATION OF THE PROPOSED METRICS

In this section we describe an experiment we have carried out for empirically validating the proposed metrics (see section 2). We should be able to draw conclusions about the relationship between the cause and the effect for which we stated a hypothesis (which we want to corroborate by means of experiments), only if the experiment is properly set up. Therefore, we have followed some suggestions provided by Wholin et al. (2000), Perry et al. (2000) and Briand et al. (1999) about how to perform controlled experiments.

To perform an experiment, several steps have to be taken and they have to be in a certain order. The experiment process can be divided into the following main activities:

4.1 Definition

As Wholin et al. (2000) suggested, we follow the GQM template (Basili and Weiss, 1984; Basili and Rombach , 1988; Van Solingen and Berghout, 1999) for goal definition. This results in the following goal:

Analyse	*UML class diagrams complexity metrics*
For the purpose of	*Evaluating*
With respect to	the correlation with maintainability sub-characteristics
From the point of view of	*researchers*
In the context of	*M.Sc. students and professors of the Engineering Software Area in the Department of Computer Science in the University of Castilla-La Mancha.*

4.2 Planning

After the definition of the experiment, the planning took place. The definition determines the foundation of the experiment -*why* the experiment is conducted- while the planning prepares for *how* the experiment is conducted.

4.2.1 Context selection

The context of the experiment is a group related to the area of Software Engineering. at the university, and hence the experiment is run-off line (not industrial software development), it is conducted by 7 professors and 10 students enrolled in the final-year of Computer Science in the Department of Computer Science at the University of Castilla-La Mancha in Spain. All of the professors belong to the Software Engineering area.

The experiment is specific since it is focused on UML class diagram structural complexity metrics. The ability to generalise from this specific context is further elaborated below when discussing threats to the experiment. The experiment addresses a real problem the correlation between metrics and maintainability sub-characteristics.

4.2.2 Hypothesis formulation

We wish to test the hypothesis that there is a significant correlation between the current metric data set (NC, NA, NM, NAssoc, NAgg, NDep, NGen, NAggH, NGenH, MaxHAgg, MaxDIT) and the subject's rating of three maintainability sub-characteristics, such as understandability, analysability and modifiability.

4.2.3 Variables selection

The independent variable is the UML class diagram structural complexity

The dependent variables are three maintainability sub-characteristics: understandability, analysability and modifiability.

4.2.4 Selection of subjects

The subjects are chosen for convenience, i.e. the subjects are students and professors that have experience in the design and development of OOIS.

4.2.5 Experiment design

We selected a within-subject design experiment, i.e. all the tests were solved by the same group of subjects. The tests were put in a different order for each subject.

4.2.6 Instrumentation

The objects were class diagrams done using UML.

The independent variable was measured through the metrics, presented in section 2.

The dependent variables were measured according to subject's rating.

4.2.7 Validity evaluation

We will discuss the empirical study's various threats to validity and the way we attempted to alleviate them:

- THREATS TO CONSTRUCT VALIDITY. We propose subjective metrics for measuring each of the dependent variables (maintainability sub-characteristics) based on the judgement of the subjects (see section 3.2). As the subjects involved in this experiment have medium experience in UML class diagram design we think their ratings can be considered significant. The independent variables (each of the metrics proposed in section 2) that measure the structural complexity of class diagrams can also be considered constructively valid, because from a system theory point of view, a system is called complex if it is composed of many (different types of elements), with many (different types of) (dynamically changing) relationships between them (Poels and Dedene, 2000a).
- THREATS TO INTERNAL VALIDITY. Seeing the results of the experiment we can conclude that empirical evidence of the existing relationship between the independent and the dependent variables exists. We have tackled different aspects that could threaten the internal validity of the study, such as: differences among subjects, knowledge of the universe of discourse among class diagrams, accuracy of subject responses, learning effects, fatigue effects, persistence effects and subject motivation.
- THREATS TO EXTERNAL VALIDITY. Two threats to external validity have been identified which limit the ability to apply any such generalisation, and we have tried to alleviate them: materials and tasks, and subject selection. In general in order to extract a final conclusion that can be generalised, we need to replicate this experiment with a greater number of subjects, including practitioners.

4.3 Operation

4.3.1 Preparation

By the time the experiment was done all of the students had had two courses on Software Engineering, in which they learnt in depth how to build OO software using UML. All the selected professors had enough experience in the design and development of OOIS. Moreover, subjects were given an intensive training session before the experiment took place.The subjects were not aware of what aspects we intended to study. Neither they were aware of the actual hypothesis stated.

We prepared the material we had to give to the subjects, consisting of 28 class diagrams of the same universe of discourse, related to Bank Information Systems. Each diagram has a test enclosed which includes the description of maintainability sub-characteristics, such as: understandability, analysability, modifiability. Each subject has to rate each sub-characteristic using a scale consisting of seven linguistic labels. For example for understandability we proposed the following linguistic labels:

Extremely difficult to understand	Very difficult to understand	A bit difficult to understand	Neither difficult nor easy to understand	Quite easy to understand	Very easy to understand	Extremely easy to understand

We also prepared a debriefing questionnaire. This questionnaire included (i) personal details and experience, (ii) opinions on the influence of different components of UML class diagrams, such as: classes, attributes, associations, generalisations, etc... on their maintainability.

4.3.2 Execution

The subjects were given all the material described in the previous section. We explained to them how to carry out the experiment. 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.

We collected all the data, including subjects' rating obtained from the responses of the experiment and the metrics values automatically calculated by means of a metric tool we had designed.

4.3.3 Data Validation

All tests were considered valid because all of the subjects have at least medium experience in building UML class diagrams and developing OOIS.

4.4 Analysis and Interpretation

As we have said before, our goal is to ascertain if any correlation exists between each of the proposed metrics and three of the maintainability sub-characteristics: understandability, analisability and modifiability.

Spearman's correlation was used to determine the correlation of the data collected in the experiment,(see table 2).

	NC	NA	NM	NAssoc	NAgg	NDep	NGen	NAggH	NGenH	MaxHagg	MaxDIT
Understandability	0.961	0.941	0.929	0.753	0.813	0.518	0.876	0.714	0.902	0.728	0.749
Analysability	0.966	0.940	0.916	0.733	0.822	0.534	0.868	0.720	0.921	0.722	0.738
Modifiability	0.950	0.924	0.908	0.733	0.818	0.522	0.865	0.719	0.888	0.725	0.751

Table 2. Spearman's correlation between UML class diagrams structural complexity metrics and understandability, analysability and modifiability

Analysing the Spearman's correlation coefficients shown in table 4, we can conclude that there exists a high correlation between most of the UML class diagram structural complexity metrics and the subject's rating of understandability, analysability and modifiability. We can deduce this due to the fact that almost all the metrics have a correlation greater than 0.7. NDep is the only one that has a lesser correlation. This fact should be studied in detail by carrying out further experimentation.

4.5 Presentation and package

The last activity is concerned with presenting and packaging of the findings. The diffusion of the experimental results and the way they are presented are relevant so that they are really put into

use. Therefore we published our findings in this paper, and we are also planning to publish a lab package on the web for replication purposes.

5. CONCLUSIONS AND FUTURE WORK

In this paper we have presented different, which are defined to assess the structural complexity of UML class diagrams obtained at high level design stage. With the objective of corroborating that there exists a great correlation between these metrics values and the maintainability of a class diagram, we have carried out a controlled experiment. Analysing the data collected using Spearman 's correlation we have concluded that most of the proposed metrics are highly correlated with the maintainability characteristics such as: understandability, analysability and modifiability.

Nevertheless, despite the promising nature of the obtained results, towards of seeking correct OO metrics applied at a high level design stage, we are aware that we need to do more metric validation, both empirical and theoretical in order to obtain conclusive evidence of the usefulness of the proposed metrics.

Pending is the theoretical validation of the proposed metrics using the DISTANCE framework proposed by Poels and Dedene (1999; 2000b), which is in our knowledge the most appropriate for OO measurements.

Regarding empirical validation we are refining this experiment in order to replicate it. For example we have found out that it is not necessary to include 28 diagrams, but it would be possible to take only the most representative. We are also designing a new experiment, in which we will give the subjects several class diagrams and some new requirements to be added. In this case the independent variable will be measured by the time spent in modification tasks, which is more objective than subjects' rating.

We also need "real data" about UML class diagram maintainability efforts, such as time spent in maintenance tasks in order to predict data that can be highly fruitful to software designers and developers. However the scarcity of such data continues to be a great problem which we must tackle to validate metrics. Brito e Abreu et al. (1999) suggested the necessity of a public repository of measurement experiences, which we think could be a good step towards achieving success in all the work done related to software measurement.

Once the proposed metrics are refined (i.e. validated or discarded) we have the plan to embed them into an OO CASE tool, for helping OO designers to take better decisions in their design tasks, which is the most important goal of any measurement proposal that aims to be useful (Fenton and Neil, 2000).

In future work, we will also tackle the measurement of other quality factors like those proposed in the ISO 9126 (1999), which not only addresses class diagrams, but also evaluates other UML diagrams, such as use-case diagrams, state diagrams, etc. To our knowledge, little work has been done towards measuring dynamic and functional models (Poels and Dedene, 2000a).

Acknowledgements

This research is part of the DOLMEN project supported by CICYT (TIC 2000-1673-C06-06).

References

- Anderson, J.R. (1983). *The Architecture of Cognition.* Cambridge, MA: Harvard Universitiy Press.
- Basili V. and Weiss D. (1984). A Methodology for Collecting Valid Software Engineering Data, *IEEE Transactions on Software Engineering,* 10, 728-738.
- Basili V. and Rombach H. (1988). The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering* 14, 758-773.
- Basili V., Shull F. and Lanubile F. (1999). Building knowledge through families of experiments. *IEEE Transactions on Software Engineering*, 25(4), 435-437.
- 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).
- Brito e Abreu F. and Carapaçua R. (1994). Object-Oriented Software Engineering: Measuring and controlling the development process. *4th Int Conference on Software Quality*, Mc Lean, Va, USA.
- Brito e Abreu F., Zuse H., Sahraoui H. and Melo W. (1999). Quantitative Approaches in Object-Oriented Software Engineering. *Object-Oriented technology: ECOOP'99 Workshop Reader*, Lecture Notes in Computer Science 1743, Springer-Verlag, 326-337.
- Calero C., Piattini M. and Genero M. (2001). Metrics for controlling database complexity. In *Developing Quality Complex Databases*. Shirley Becker Ed. Idea Group Publishing
- Chidamber S. and Kemerer C. (1994). A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*. 20(6), 476-493.
- Fayyad U., Piatetsky-Shapiro G. and Smyth P. (1996). The KDD Process for Extracting Useful Knowledge from Volumes of Data. *Communications of the ACM*, 39(11), 27 – 34.
- Fenton N. and Neil, M. (2000). Software Metrics: a Roadmap. *Future of Software Engineering.* Ed:Anthony Finkelstein, ACM, 359-370.
- Fenton N. and Pfleeger S. (1997). *Software Metrics: A Rigorous Approach.* 2nd. edition. London, Chapman & Hall.
- Genero, M., Piattini, M. and Calero, C. (2000). Early Measures For UML class diagrams. *L'Objet.* 6(4), Hermes Science Publications, 489-515.
- ISO/IEC 9126-1.2. (1999). Information technology- Software product quality – Part 1: Quality model.
- Henderson-Sellers B. (1996). *Object-Oriented Metrics - Measures of complexity*. Prentice-Hall, Upper Saddle River, New Jersey.
- Kitchenham, B., Pflegger, S. and Fenton, N. Towards a Framework for Software Measurement Validation. *IEEE Transactions of Software Engineering*, 21(12), (1995) 929-943.
- Lorenz M. and Kidd J. (1994). *Object-Oriented Software Metrics: A Practical Guide.* Prentice Hall, Englewood Cliffs, New Jersey.

- Marchesi M. (1998). OOA Metrics for the Unified Modeling Language. *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering*, 67-73.
- Melton, A. (ed.) (1996). *Software Measurement.* London. International Thomson Computer Press.
- Object Management Group. (1999). *UML Revision Task Force. OMG Unified Modeling Language Specification, v. 1.3. document ad/99-06-08.*
- Olivas J. A. and Romero F. P. (2000). FPKD. Fuzzy Prototypical Knowledge Discovery. Application to Forest Fire Prediction. *Proceedings of the SEKE'2000*, Knowledge Systems Institute, Chicago, Ill. USA, 47 – 54.
- Olivas J. A. (2000). Contribution to the Experimental Study of the Prediction based on Fuzzy Deformable Categories, PhD Thesis, University of Castilla-La Mancha, Spain.
- Perry D., Porte A. and Votta L. (2000). Empirical Studies of Software Engineering: A Roadmap. *Future of Software Engineering.* Ed. Anthony Finkelstein, ACM, 345-355.
- Poels G. and Dedene G. (1999). DISTANCE: A Framework for Software Measure Construction, research report DTEW9937, Dept. Applied Economics, Katholieke Universiteit Leuven, Belgium, 46 p.
- 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, (2000a), 499-512.
- Poels G. and Dedene G. (2000b). Distance-based software measurement: necessary and sufficient properties for software measures. *Information and Software Technology*, 42(1), 35-46.
- Schneidewind N. (1992). Methodology For Validating Software Metrics. *IEEE Transactions of Software Engineering*, 18(5), 410-422.
- Siau, K. (1999). Information Modeling and Method Engineering: A Psychological Perspective. *Journal of Database Management* 10 (4), 44-50.
- Van Solingen R. and Berghout E. (1999). *The Goal/Question/Metric Method: A practical guide for quality improvement of software development.* McGraw-Hill.
- Wohlin C., Runeson P., Höst M., Ohlson M., Regnell B. and Wesslén A. (2000) *Experimentation in Software Engineering: An Introduction.* Kluwer Academic Publishers.
- Zadeh L. (1982). A note on prototype set theory and fuzzy sets. *Cognition* 12, 291 – 297.
- Zuse H. (1998). *A Framework of Software Measurement.* Berlin, Walter de Gruyter.