5th EAST-EUROPEAN CONFERENCE

ADBIS'2001

ON ADVANCES IN DATABASES AND INFORMATION SYSTEMS

RESEARCH COMMUNICATIONS

Edited by
Albertas Čaplinskas
Johann Eder

Vilnius
Lithuania
September 25-28, 2001

Vol. 1

Advances in Databases and Information Systems. 5ht East-European Conference ADBIS' 2001. Research Communications. Vol. 1.
Albertas Čaplinskas, Johann Eder (eds.). Vilnius: Technika, 2001. 244 p.: iliustr.

This book contains research communications, professional communications, and tutorials of the 5th East-European Conference on Advances in Databases and Information Systems. The book is published in three volumes. The first volume includes the final versions of 19 research papers presented on regular sessions of the Conference. The papers span a wide spectrum of the database and information systems field: from query optimisation, and transaction processing via design methods to application oriented topics like XML and data on the web. They are related to European Commission's initiative "eEurope - Information Society for All" and represent research areas that will greatly influence the functionality, usability and acceptability of future information products and services.

Cover design: Rigimantas Gedgaudas

VGTU leidyklos „Technika" 670 mokslo literatūros knyga

ISBN 9986-05-449-4 (1 tomas)
ISBN 9986-05-452-4 (3 tomai)

Table of Contents

Temporal Information Systems. Reengineering of Legacy Systems

Information Systems Design

A Prediction Model for OO Information System Quality Based on Early Indicators

Marcela Genero [1], Luis Jiménez [2], Mario Piattini[1]

[1]Grupo ALARCOS
[2]Grupo ORETO
University of Castilla-La Mancha
Ronda de Calatrava, 5
13071, Ciudad Real (Spain)
E-mail: {mgenero, ljimenez, mpiattin}@inf-cr.uclm.es

Abstract. Conceptual modelling is a key task in the early phases of an information system (IS) life cycle. In the development of object-oriented information systems (OOIS) class diagrams become the conceptual schema that reflects not only the objects of the application domain but also the behaviour of them. They lay the foundation of all later design and implementation work. Hence class diagram quality is a crucial issue that must be evaluated (and improved if necessary) in order to get getting quality OOIS, which is the main concern of present day software development organisations. It is in this context where software measurement plays an important role, because the early availability of metrics could contribute to build better OOIS. After a thorough review of the existent OO measures applicable to class diagrams at high level design stage, we have presented in [13] a set of metrics for the structural complexity of class diagrams built using the Unified Modelling Language (UML). The main goal of this work is the empirical validation of those metrics. We will present a controlled experiment carried out to assess the capability of those metrics to be used as early maintainability indicators. Using the data collected in the experiment we will build a maintainability prediction model for class diagrams based on fuzzy classification and regression trees.

Keywords: conceptual modelling, object-oriented information systems maintainability, metrics, class diagrams, structural complexity, UML, empirical validation, fuzzy classification and regression trees, prediction models

1 Introduction

Conceptual modelling is a key task in the early phases of an information system (IS) life cycle. Moreover, modern approaches towards OO system development, like Catalysis [6] and Rational Unified Process [27], have included conceptual modelling as a relevant task. In the development of object-oriented information systems (OOIS) class diagrams become the conceptual schema that reflects not only the objects of the application domain but also the behaviour of them. They lay the foundation of all later design and implementation work. Hence class diagram quality is a crucial issue that

must be evaluated (and improved if necessary) in order to get quality OOIS, which is the main concern of present day software development organisations.

The early focus on class diagram quality may help IS designers build better OOIS, without unnecessary rework at late stages of the development when any changes are more expensive and more difficult to perform. It is in this context where software measurement plays an important role, because the early availability of metrics could contribute to evaluate class diagram quality in an objective way avoiding bias in the quality evaluation process.

Most of the existing literature about OO measures [16], [22], [33] is related to measures which can only be applied once a software product is completed or nearly complete, and these provide information too late in order to lead us to build quality OOIS.

After a thorough review of some of the existing OO measures, applicable to class diagrams at high level design stage [5], [20], [3], [21] we have proposed [13] a set of measures which measures the structural complexity of class diagrams due to the usage of relationships (associations, generalisations, aggregations and dependencies). Those measures are applicable to class diagrams built using the Unified Modelling Language (UML) at a high level design stage. However, the simple proposal of metrics is of no value if its practical use is not demonstrated empirically by experimentation, either by means of case studies taken from real projects or by controlled experiments. Therefore, empirical validation is crucial for the success of any software measurement project [18], [11], [29], [1].

Given that maintenance was (and will continue to be) the major resource waster in the whole software life cycle, maintainability has became one of the software product quality characteristic [17] that software development organisations are more worried about. As an external quality attribute, maintainability can only be measured when the product is finished, so our idea is to use the early metrics we proposed [13] as early maintainability indicators. Therefore, we will present a controlled experiment carried out to assess the capability of those metrics to be used as early maintainability indicators for class diagrams. From the empirical data we will build a maintainability prediction model for class diagrams, using a novel approach based on fuzzy classification and regression trees.

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 proposed in [13]. In section 3 we will describe each of the steps followed to perform a controlled experiment, with the objective of empirically validating those metrics. In section 4 we will show how to predict class diagram maintainability using a prediction model based on fuzzy classification and regression trees. Lastly, section 5 summarises the paper, draws our conclusions, and presents future trends in metrics for object modelling using UML.

2 A proposal of metrics for UML class diagrams

In this section we present some of the metrics proposed in [13] and some other typical OO measures, which can be applied to class diagrams at a high level design stage. We

only consider those related to the class diagram as a whole, which we called "Class Diagram-Scope metrics".

- NUMBER OF CLASSES. (NC) is the total number of classes within a class diagram.
- NUMBER OF ATTRIBUTES. (NA) is the total number of attributes within a class diagram.
- NUMBER OF METHODS. (NM) is the total number of methods within a class diagram.
- NUMBER OF ASSOCIATIONS. (NAssoc) is defined as the total number of associations within a class diagram.
- NUMBER OF AGGREGATION. (NAgg) is defined as the total number of aggregation relationships within a class diagram (each whole-part pair in an aggregation relationship).
- NUMBER OF DEPENDENCIES. (NDep) is defined as the total number of dependency relationships within a class diagram.
- 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) is defined as the total number of generalisation hierarchies in a class diagram
- MAXIMUM DIT. The Maximum DIT in a class diagram 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.

3 Empirical validation of the proposed metrics

Taking into account some suggestions provided in [2], [24] about how to do empirical studies in software engineering, we carried out a controlled experiment pursuing the following goals:

1. To ascertain if any relationship exists between each of the proposed metrics and three of the maintainability sub-characteristics: understandability, analysability and modifiability [17] related to UML class diagrams.
2. To establish a prediction model for those maintainability sub-characteristics from metric values obtained at a high level design stage.

3.1 Subjects

The experimental subjects used in this study were: 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 and they have ample experience in the design and development of OO software. 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. Moreover, subjects were given an intensive training session before the experiment took place.

3.2 Experimental materials and tasks

The subjects were given twenty eight UML class diagrams of the same universe of discourse, related to Bank Information Systems. Each diagram has a test enclosed which includes the description of three 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 use the linguistic labels shown in table 1.

Table 1. Understandability 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 allowed one week to do the experiment, i.e., each subject had to 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 class diagrams, such as: classes, attributes, associations, generalisations, etc... on their maintainability.

3.3 Experimental design and data Collection

The INDEPENDENT VARIABLE is the structural complexity of UML class diagram measured with the metrics proposed in sections 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 the subjects had 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 Data Analysis and Results

Due to the nature of the software development process and products, one cannot expect to use in Software Engineering the same measurement data analysis techniques that are used in "exact" sciences, e.g., Physics, Chemistry, nor obtain the same degree of precision and accuracy [23]. The calculation of correlation or the use of regression analysis is not sufficient [4]. Statistic and numerical machine learning techniques are

very dependant on the set of data and the models and are not qualitative [28]. Therefore, we need a machine learning technique that allow us to build prediction models with two characteristics: they must be highly qualitative and more straightforward and intelligible to human beings.

Following this idea, [9] and [10] proposed a fuzzy classification using a set of software quality metric values by an unsupervised learning process. However our proposal is different because we follow a supervised learning method, which is a novel data analysis approach based on a rule system with linguistic variables [32]. This approach, as a method of supervised learning, provides models that allow us to discover the most relevant conceptual relationships between the data we are analysing, where the accuracy of those models is sacrificed in favour of its simplicity and easiness to understand.

This method for induction of fuzzy rule system is a generalisation of the classical regression approach. For the sake of brevity we only show the results obtained by the application of this technique. More information about it could be found in [14], [19], [7].

First, we establish fuzzy rule systems for understandability, analysability and modifiability. For example for understandability we use a learning set with X={set of our metrics} and Y={the values of understandability obtained in our experiment}. We have obtained 476 values (28 class diagrams and 17 subjects) of this unknown function:

F1(NC,NA,NM,NAssoc,NAgg,NAggVC,NAggH,NDepR,NDepVC,NGen,NGH,MaxDIT)=understandability

We also consider functions for the analysability and modifiability, as is shown below:

F2(NC,NA,NM,NAssoc,NAgg,NAggVC,NAggH,NDepR,NDepVC,NGen,NGH, MaxDIT)=analisability

F3(NC,NA,NM,NAssoc,NAgg,NAggVC,NAggH,NDepR,NDepVC,NGen,NGH, MaxDIT)=modifiability

By the induction method described above we have obtained a prediction model composed of fuzzy rules, generated from the obtained fuzzy rules. A fuzzy rule is formed by the antecedent part (left part of the rule) and the consequent part (right part of the rule) to reflect causal-effect relation. The antecedent part is formed by aggregation of fuzzy statements as " X is A", where X is a metric value and A is a fuzzy set over metric domain.

In this example we have used a trapezoidal function for fuzzy sets, which are defined by four numbers. The fuzzy set [a,b,c,d] has the following membership:

$$A(x,a,b,c,d) = \begin{cases} 0 & x <= a \\ \dfrac{x-a}{b-a} & a < x < b \\ 1 & b \le x \le c \\ \dfrac{d-x}{d-c} & c < x < d \\ 0 & x \ge d \end{cases}$$

As our goal is to define a rule system with linguistic variables we briefly introduce the concept of linguistic variables. A linguistic variable [32] is a tuple (X, T(X), V, G, M) where X is the variable name (in our case metric names), T(X) is the label set, V is the domain in which the variables are defined, M is a semantic rule that associate to each label a fuzzy set defined over the domain V.

With our method once the fuzzy sets are inducted, we will label each fuzzy set (see table 2) for establishing the linguistic variables we will use. In table 2 only appear those metrics that have bee resulted relevant for the maintainability sub-characteristics we have considered.

Table 2. Labels of fuzzy sets.

Metrics	Very_Low (V_L)	Low (L)	Medium (M)	High (H)	Very_High (V_H)
NC	[2,2,3,4]	[3,4,5,7]	[5,7,11,17]	[11,17,20,22]	[20,22,29,29]
NA		[4,4,22,30]	[22,30,34,41]	[34,41,56,56]	
NM	[8,8,21,30]	[21,30,30,35]	[30,35,38,65]	[38,65,69,73]	[69,73,98,98]
NGen		[0,0,7,16]		[7,16,24,24]	
NAgg		[0,0,2,4]		[2,4,9,9]	

Table 3 shows the set of fuzzy rules of the understandability model (they were generated using an automatic tool). The rows named 1-15 represent the 15 rules obtained by the induction approach where Everything (E) is the complete domain of the metric. We can read the rule 14 thus ***"IF NC is Very_High and NA is High and NM is High or Very_High and NAgg is Low and NGen is High THEN Understandability is 5.6624 "*** . This rule makes 0.009 error and has a data coverage percentage of 0.68%.

Table 3. Understandability qualitative linguistic fuzzy model.

	NC	NA	NM	NAgg	NGen	Unders-tandability	ERROR	COV%
1	V_L	E	E	E	E	1.6661	0.1104	19.6956
2	L	E	E	E	E	2.1647	0.1605	14.7674
3	M	E	L	E	E	2.7241	0.0964	7.1446
4	M	E	V_L	E	E	2.908	0.0828	6.6071
5	M	E	M	E	E	3.3291	0.1902	10.4751
6	M	E	H or V_H	E	E	3.7804	0.0893	3.5239
7	H	E	Not (H	E	E	4.1179	0.0668	2.8853

	NC	NA	NM	NAgg	NGen	Unders-tandability	ERROR	COV%
			or V_H)					
8	H	E	H	E	E	4.714	0.1275	6.625
9	V_H	E	Not (H or V_H)	E	E	4.7853	00084	0.2492
10	H	E	V_H	E	E	5.0307	01272	5.8916
11	V_H	H	H or V_H	E	L	5.3867	0.0166	1.2598
12	H	E	V_H	E	E	5.4917	0.0637	3.5325
13	V_H	M	H or V_H	E	E	5.6533	0.0189	1.2511
14	V_H	H	H or V_H	L	H	5.6624	0.0095	0.6839
15	V_H	H	H or V_H	H	H	5.7305	0.1837	14.8324

Where: the first column represents each rule number; the columns **NC, NA, NM, NAgg** and **NGen** are the linguistic variables associated with each metric name; the column **Understandability** is the output or the consequent of the rules; the column **ERROR** is the error produced when the rule is generated and the column **COV%** is the data coverage percentage taking into account the sample data.

By our approach we also have obtained linguistic models for analysability and modifiability, which are shown in tables 4 and 5. Those can be interpreted in the same way as table 3.

Table 4. Analysability qualitative linguistic fuzzy model.

	NC	NA	NM	NGen	Analysa-bility	ERROR	COV%
1	V_L	E	E	E	1.828	0.1299	19.6956
2	L	E	E	E	2.3444	0.1723	14.7674
3	M	E	L	E	2.9064	0.0972	7.1446
4	M	E	V_L	E	2.9739	0.0825	6.6071
5	M	Not L	M	E	3.0203	0.032	2.0865
6	M	L	M	E	3.413	0.1735	8.3886
7	M	E	H or V_H	E	3.9167	0.0897	3.5239
8	H	E	Not (H or V_H)	E	4.163	0.0585	2.8853
9	H	E	H	E	4.8479	0.1242	6.625
10	V_H	E	Not (H or V_H)	E	4.8697	0.0078	0.2492
11	H	E	V_H	E	5.1622	0.1049	5.8916
12	H	E	H or V_H	E	5.4117	0.0626	3.5325
13	V_H	H	H or V_H	L	5.6073	0.013	1.2598
14	V_H	M	H or V_H	E	5.6815	0.0154	1.2511
15	V_H	H	H or V_H	H	5.6925	0.1662	15.5163

Table 5. Modifiability qualitative linguistic fuzzy model.

	NC	NA	NM	Modifia-bility	ERROR	COV%
1	V_L	E	E	1.9457	0.1416	19.6956
2	L	L	E	2.3705	0.1796	14.4859

	NC	NA	NM	Modifia-bility	ERROR	COV%
3	M	E	L	2.9299	0.0901	7.1446
4	M	Not L	M	3.1397	0.035	2.0865
5	M	E	V_L	3.1471	0.0894	6.6071
6	M	L	M	3.5483	0.1911	8.3886
7	M	E	H or V_H	4.0154	0.0948	3.5239
8	H	E	Not (H or V_H)	4.2811	0.0566	2.8853
9	L	Not L	E	4.6282	0.0077	0.2814
10	H	E	H	4.905	0.1129	6.625
11	V_H	E	Not (H or V_H)	4.9974	0.0081	0.2492
12	H	E	V_H	5.2336	0.1249	5.8916
13	H	E	H or V_H	5.5931	0.0604	3.5325
14	V_H	H	H or V_H	5.7989	0.1965	16.7761
15	V_H	M	H or V_H	5.8278	0.0167	1.2511

These fuzzy rules systems shown in table 3, 4 and 5, represent prediction models for the understandability, analysability and modifiability. These models are highly natural and closer to the human mind, therefore they are in accordance with our goals, presented in the beginning of this section.

Our method also allow us to identify which are the most relevant metrics associated to each maintainability sub-characteristics, as we described in the following:

- Understandability: NC, NA, NM, NAgg and NGen metrics (see table 3).
- Analysability: NC, NA, NM and NGen metrics (see table 4).
- Modifiability: NC, NA and NM metrics (see table 5).

3.5 Threats to Validity

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

- CONSTRUCT VALIDITY. The degree to which the independent and the dependent variables accurately measure the concepts they purport to measure.
- INTERNAL VALIDITY. The degree to which conclusions can be drawn about the causal effect of independent variables on the dependent variables.
- EXTERNAL VALIDITY. The degree to which the results of the research can be generalised to the population under study and other research setting.

3.5.1 Threats to Construct Validity

The dependent variables we used are maintainability sub-characteristics: understandability, analysability and modifiability. We propose subjective metrics for them (using linguistic variables), based on the judgement of the subjects (see section 3.3). As the subjects involved in this experiment have medium experience in OOIS design and implementation we think their ratings could be considered significant.

For construct validity of the independent variables, we have to address the question to which degree the metrics used in this study measure the concept they purport to measure. Our idea is to use metrics presented in section 2. to measure the

structural complexity of an UML class diagram. 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 [26]. According to this, we think that the construct validity of our independent variables can thus be considered satisfactory. In spite of this, we consider that more experiments must be done, in order to draw a final conclusion to assure construct validity.

3.5.2 Threats to Internal Validity

The following issues have been dealt with:

- DIFFERENCES AMONG SUBJECTS. Using a within-subjects design, error variance due to differences among subjects is reduced. As Briand et al. remarks [2] in software engineering experiments when dealing with small samples, variations in participant skills are a major concern that is difficult to fully address by randomisation or blocking. In this experiment, professors and students had the same degree of experience in modelling with UML.
- KNOWLEDGE OF THE UNIVERSE OF DISCOURSE AMONG CLASS DIAGRAMS. Class diagrams were designed for the same universe of discourse, only varying the number of attributes, classes, associations, i.e. their constitutents parts. So that, the knowledge of the domain doesn't attempt to the internal validity.
- ACCURACY OF SUBJECT RESPONSES. Subjects assumed the responsibility for rating each maintainability sub-characteristics. As they have M experience in OO software design and implementation, we think their responses could be considered valid. However we are aware that not all of them have exactly the 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 in UML. One similar experiment was done by these subjects but related to entity relationship diagrams [14].
- 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 subjects 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.

3.5.3 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 H levels of industrial experience, so, experiments with students could be appropriate [1].

We want to highlight that this is a first approach to predict UML class diagram maintainability. We are aware that it is necessary to replicate this experiment with a greater number of subjects, including practitioners. It could also be useful to have "real data" about UML class diagram maintainability efforts, like time spent in maintenance tasks in order to predict data that can be highly productive for software designers and developers. However, this is the major problem confronting software measurement, the scarce availability of data related to "real projects". It would be necessary to have access to a public repository [4] with metric related data in order to reuse past experiences and learn and move on from them.

4 Prediction of class diagram maintainability

Now, by way of an example we will only demonstrate how the understandability of a class diagram can be predicted using the proposed model, presented above.

For example, suppose that we want to answer the following question,"**What is the value of Understandability, if we know that NC is 21, NA is 50, NM is 70, NAgg is 9 and NGen is 20 ?".** This question, can be answered by inference following table 6.

Table 6. Inference of a new value of understandability.

RULE	NC=21	NA=50	NM=70	NAgg=9	NGen=20	Understandability	MIN	MIN* Understandability
1	0	1	1	1	1	**1.67**	0	0
2	0	1	1	1	1	**2.16**	0	0

RULE	NC=21	NA=50	NM=70	NAgg=9	NGen=20	Understandability	MIN	MIN* Understandability
3	0	1	0	1	1	**2.72**	0	0
4	0	1	0	1	1	**2.91**	0	0
5	0	1	0	1	1	**3.33**	0	0
6	0	1	0.75	1	1	**3.78**	0	0
7	0.5	1	0.25	1	1	**4,12**	0.25	1.03
8	0.5	1	0.75	1	1	**4.71**	0.5	2.36
9	0.5	1	0.25	1	1	**4.79**	0.25	1.2
10	0.5	1	0.25	1	1	**5.03**	0.25	1.26
11	0.5	1	0.75	1	0	**5.39**	0	0
12	0.5	1	0.25	1	1	**5.49**	0.25	1.37
13	0.5	0	0.75	1	1	**5,65**	0	0
14	0.5	1	0.75	0	1	**5.66**	0	0
15	0.5	1	0.75	1	1	**5.73**	0.5	2.87
						SUM	**2**	**10.08**

Where: the column **RULE** is the rule number; the columns **NC, NA, NM, NAgg, NGen** represent the degree of membership of each; the column **Understandability** represents the output or the consequent of the rules; the column **MIN** represents minimum of the degree of membership of the antecedents and the column **MIN*Understandability** represents the contribution of each rule to the final solution 5.04.

Using approximate reasoning [30] and with a simplified consequent, detailed in table 3, we obtain the result shown in figure 1.

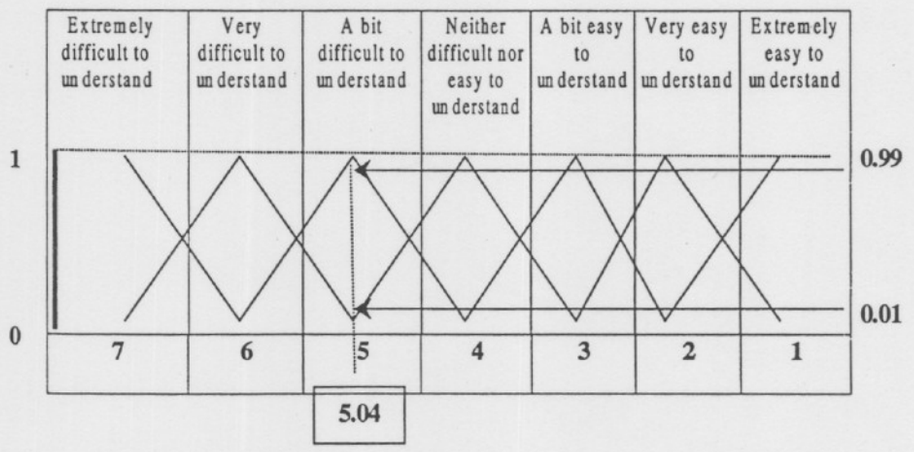

Fig. 1. Inference of a new understandability value

The value of understandability is 10.08/2=5.04. This value is 99% a bit difficult to understand and 1% very difficult to understand. This value is obtained making a weigted average of the output (maintenanse time) of each rule with your degree of membership. Even though this is a very simple example, it illustrates thatbit is possible to predict the class diagram understandability, modifiability and analy sability from the metrics values, early in the development of OOIS, helping thus the OOIS designers.

5 Conclusions and future work

It is widely recognised that the quality assurance of OOIS must be guaranteed since the early phases of its life cycle. OO conceptual models, like class diagrams are the key artifact of the early development, so focusing in their quality should contribute to the quality of the OOIS which is ultimately implemented.

We have presented a set of measures for evaluating UML class diagram structural complexity. From metrics values we have built prediction models for the understandability, analysability and modifiability of class diagrams based on fuzzy classification and regression trees. The data used to built those prediction models was collected through a controlled experiment.

That experiment also allowed us to highlight the most relevant metrics associated to each maintainability sub-characteristics, which are for understandability NC, NA, NM, NAgg, NGen, for analysability NC, NA, NM, NGen, and for modifiability NC, NA and NM. Nevertheless, given that these conclusion can be considered as preliminaries we cannot discard the others metrics. We are aware that we need to do more metric validation, specially taken data from "real cases" to derive some design heuristics that could/should be included on design tools. This could really help IS designers to take better decisions in their design tasks, which is the most important goal that must pursue any measurement proposal if it pretends to be useful [12].

We have developed another prediction model for those maintainability sub-characteristics [15] based on fuzzy deformable prototypes. As future work we will compare the prediction capability of both models applied to real data

We will also focus our research on measuring other quality factors like those proposed in the [17], which not only tackle class diagrams, but also evaluate other UML dynamic diagrams, such as use-case diagrams, state diagrams, etc. To our knowledge, little work has been done towards measuring dynamic and functional models [8]; [25]; [26]. This is an area which need further investigation [4].

Acknowledgements

This research is part of the DOLMEN project supported by CICYT (TIC 2000-1673-C06-06) and the CIPRESES project supported by CICYT (TIC 2000-1362-C02-02).

IEEE Transactions on Software Engineering, 25(4), 1999, 455-457.

2. 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.
3. Brito e Abreu, F. and Carapaçua, R. Object-Oriented Software Engineering: Measuring and controllong the development process. *4th Int Conference on Software Quality*, Mc Lean, Va, USA, 1994.
4. 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.
5. Chidamber, S. and Kemerer, C. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering*. 20(6), 1994, 476-493.
6. D´Souza, D. F. and Wills, A. C. Objects, Components and Frameworks with UML: the Catalysis Approach. Addison-Wesley, 1999.
7. Delgado, M., Gómez Skarmeta, A. and Jiménez, L. International Journal of Intelligent Systems, 16, 2001, (to appear).
8. Derr, K. Applying OMT. SIGS Books, New York, 1995.
9. Ebert, C. Rule-based fuzzy classification for software quality control, *Fuzzy Sets and Systems*, 63, 1993, 349-358.
10. Ebert, C. and Baisch, E. Metrics for identifying critical components in software projects, Chapter in: Handbook of Software Engineering and Knowledge Engineering, Vol. 1, 2001 (to appear).
11. Fenton, N. and Pfleeger, S. Software Metrics: A Rigorous Approach. 2nd. edition. London, Chapman & Hall, 1997.
12. Fenton, N. And Neil, M. Software Metrics: a Roadmap. Future of Software Engineering. Ed:Anthony Finkelstein, ACM, 2000, 359-370.
13. Genero, M., Piattini, M. and Calero, C. Early Measures For UML class diagrams. *L´Objet*. 6(4), Hermes Science Publications, 2000, 489-515.
14. Genero, M., Jiménez, L. and Piattini, M. Measuring the quality of entity relationship diagrams. *Proceedings of the 19th International Conference on Conceptual Modeling (ER 2000)*, Salt Lake City, 2000, 513-526.
15. Genero, M., Olivas, J., Piattini, M. and Romero, F. Using metrics to predict OO information systems maintainability. *CAISE 2001*, Interlaken Switzlerland, 2001, (to appear).
16. Henderson-Sellers, B. Object-Oriented Metrics - Measures of complexity. Prentice-Hall, Upper Saddle River, New Jersey, 1996.
17. ISO/IEC 9126-1.2. Information technology- Software product quality – Part 1: Quality model, 1999.
18. Kitchenham, B., Pflegger, S. and Fenton, N. Towards a Framework for Software Measurement Validation. *IEEE Transactions of Software Engineering*, 21(12), 1995, 929-943.
19. Linares, L. J., Delgado, M. and Skarmeta, A. Regression by fuzzy knowledge bases. Proceedings of the *4th European Congress on Intelligent Techniques and Soft Computing*. Aachen, Germany, September, 1996, 1170-1176.
20. Lorenz, M. and Kidd, J. *Object-Oriented Software Metrics: A Practical Guide*. Prentice Hall, Englewood Cliffs, New Jersey, 1994.

21. Marchesi, M. OOA Metrics for the Unified Modeling Language. *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering*, 1998, 67-73.
22. Melton , A. (ed.). Software Measurement. London, International Thomson Computer Press, 1996.
23. Morasca, S. and. Ruhe, G. Guest Editors "Introduction: Knowledge Discovery From Empirical Software Engineering Data", *International Journal of Software Engineering and Knowledge Engineering*, 9 (5), 1999, 495-498.
24. Perry, D., Porte, A. and Votta, L. Empirical Studies os Software Engineering: A Roadmap. *Future of Software Engineering*. Ed:Anthony Finkelstein, ACM, 2000, 345-355.
25. 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.
26. Poels, G. and Dedene, G. Measures for Assessing Dynamic Complexity Aspects of Object-Oriented Conceptual Schemes. *Proceedings of the 19th International Conference on Conceptual Modeling (ER 2000)*, Salt Lake City, October, 2000, 499-512.
27. Rational Software. Object Oriented Analysis and Design, Student Manual. http://www.rational.com/, 1998.
28. Sahraoui, H., Boukadoum, M. and Lounis, H. Using Fuzzy Threshold Values for Predicting Class libraries Interface Evolution. *Proceedings of 4th International ECCOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, June 13, Cannes, France, 2000.
29. Schneidewind, N. Methodology For Validating Software Metrics. *IEEE Transactions of Software Engineering*, 18(5), 1992, 410-422.
30. Sugeno, M. An Introductory Survey of Fuzzy Control. *Information Sciences*, 36, 1985, 59-83.
31. Zadeh, L. Fuzzy sets. *Information and control*, 1965, 338-353.
32. Zadeh, L. The Concept of Linguistic Variable and its Applications to Approximate Reasoning Part I. *Information Sciences*, 8, 1973, 199-249.
33. Zuse, H. *A Framework of Software Measurement*. Berlin, Walter de Gruyter, 1998.