Empirical Validation of Class Diagram Complexity Metrics

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

Luis Jiménez
ORETO Research Group
Department of Computer Science
University of Castilla-La Mancha
Ronda de Calatrava, 5
13071, Ciudad Real (Spain)
E-mail: ljimenez@inf-cr.uclm.es

Abstract

One of the principal objectives of software engineering is to improve the quality of software products. It is widely recognised that the quality assurance of software products must be guaranteed from the early phases of development. As a key artifact produced in the early development of object oriented information systems (OOIS), class diagram quality has a great impact on the quality of the software product which is finally delivered. 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. After a thorough review of the existing OO measures applicable to class diagrams at a high level design stage, we have presented in Genero et al. (2000a) a set of metrics for the structural complexity of class diagrams built using the Unified Modelling Language (UML). We focus on class diagram structural complexity, an internal quality attribute, which we believe could be closely correlated with one of the most critical external quality attributes, such as class diagram maintainability. As the main goal of this paper is the empirical validation of those metrics, we will present two controlled experiments carried out to corroborate if those metrics are closed to class diagram maintainability and thus could be used as early maintainability indicators. Based on data collected in the experiments we will build a prediction model for class diagram maintainability using a method for induction of fuzzy rules.

Keywords. Software quality, structural complexity metrics, class diagram maintainability, empirical validation, prediction model

1. Introduction

A great effort has been made in the field of software measurement in order to achieve better quality OOIS [1, 2,3,4], 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 [5], like for example maintainability.

As class diagrams constitute a key artifact in the early development of OOIS, the effort expended on improving their quality is likely to pay off many times over in later phases. It is in this context where software measurement plays an important role, because the early availability of metrics could contribute to evaluating class diagram quality in an objective way avoiding bias in the quality evaluation process.

In response to the great demand for measures for measuring quality characteristics of class diagrams, such as maintainability and after a thorough review of some of the existing OO measures that can be applied at a high level design stage [6,7,8,9], we have proposed a set of measures for UML class diagram structural complexity in [10]. We focused on UML because it is considered a standard in OO modelling [11]. As maintainability is an external quality characteristic that can be evaluated once a product is finished or nearly finished, we centre 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 life-cycle.

Empirical validation is critical for the success of any measurement activity [3,12,13,14], because by means of

1522-4902/01 $17.00 © 2001 IEEE

empirical validation we can demonstrate with real evidence that the measures we proposed serve the purpose they were defined for and that they could be fruitful in practice. As our goal is to demonstrate the empirical validity of the metrics we proposed [10], we will present two controlled experiments carried out to assess if a close relation exists between the metrics and class diagram maintainability. If that is the case, those metrics can be used as early maintainability indicators for class diagrams. From the empirical data collected in the experiments we will build a prediction model for class diagram maintainability, using a novel approach based on fuzzy rules.

The objective of this paper is threefold:

1. To present metrics for UML class diagram structural complexity (see section 2)
2. To show two controlled experiments we have carried out in order to evaluate if there is empirical evidence that UML class diagram structural complexity metrics are related to class diagram maintainability (see section 3).
3. To use the empirical data collected in the experiments for building a prediction model for class diagram maintainability. To build the prediction model we use a method for induction of fuzzy rules (see section 3 and 4).

The last section presents some concluding remarks and identifies further work related to metrics for object-oriented conceptual modelling using UML.

2. A proposal of metrics for UML class diagrams

We only present here those metrics proposed in [10] 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.

Table 1. Metrics for UML class diagram structural complexity

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)	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 generalisation 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.

3. Empirical validation of the proposed metrics

In this section we describe two experiments we have carried out for empirically validating the proposed metrics (see section 2). We will only 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), if the experiment is properly set up. Therefore, we have followed some suggestions provided by Wholin et al. [15], Perry et al. [16] and Briand et al. [17] about how to perform controlled experiments.

In the remainder of this section we explain how we have performed each of the experiments. The second experiment is a replication of the first, changing the subjects who performed it.

3.1 Definition

As Wholin et al. [15] suggested, we follow the GQM template [18,19] for goal definition.

The goal of the first experiment is shown in table 2.

Table 2. Goal of the first experiment

Analyse	UML class diagram structural complexity
For the purpose of	Evaluating
With respect to	The capability of being used as early quality indicators
From the point of view	OOIS designers
In the context of	M.Sc. students in the final year of Computer Science and professors of the Area of Software Engineering at the Department of Computer Science in the University of Castilla-La Mancha

The goal of the second experiment is the same, changing the part of the context, which is M.Sc. students enrolled in the third year of Computer Science in the University of Castilla-La Mancha.

3.2 Planning

After the definition of the experiments -*why* the experiment is conducted-, the planning took place. The planning prepares for *how* the experiment is conducted, including the following activities:

- **Context selection.** The context of the first 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 was conducted by 10 professors and 20 students (we called them "expert subjects") 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 second experiment was conducted by 22 students enrolled in the third year of Computer Science in the same Department (we called them "novice subjects"). The experiments are 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 experiments address a real problem, the relationship between metrics and the time spent in maintenance tasks.
- **Hypothesis formulation.** An important aspect of experiments is to know and to state in a clear and formal fashion what we intend to evaluate in the experiment. This leads us to the formulation of a hypothesis (or several hypotheses). We wish to test the hypothesis "A close relationship exists between the current metric data set (NC, NA, NM, NAssoc, NAgg, NDep, NGen, NAggH, NGenH, MaxHAgg, MaxDIT) and the UML class diagram maintainability".
- **Variables selection.** The independent variable is the UML class diagram structural complexity. The dependent variable is UML class diagram maintainability.
- **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.
- **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
- **Instrumentation.** The objects were class diagrams done using UML. The independent variable was measured through the metrics, presented in section 2. The dependent variable was measured by the time spent by the subjects on modifying each diagram in order to satisfy the new requirements. We called this time "maintenance time".

3.3 Operation

It is in this phase where measurements are collected, including the following activities:

- **Preparation.** By the time the experiments were done the "expert subjects" had had two courses on Software Engineering and the "novice experts" one. In those courses they learnt 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 were they aware of the actual hypothesis stated.
  We prepared the material we had to give to the subjects, consisting of nine UML class diagrams of different application domains, general enough to be easily understood by each of the subjects. The diagrams have different complexity, considering a broad range of metrics values. Each diagram included a brief specification about what the diagram represents and two new requirements. Each subject has to modify the class diagrams according to the

new requirements and to write down the time spent in performing those modifications.

- **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 to carry out the test alone, and could use unlimited time to solve it. We collected all the data including, the modified class diagrams with the maintenance time obtained from the responses of the experiments and the metrics values automatically calculated by means of a metric tool we had designed.
- **Data Validation.** Once the data was collected, we controlled if the tests were complete and if the modifications were well done. We discarded seven tests in the first experiment and three tests in the second. So we consider the responses of 23 subjects in the first experiment, and 19 in the second.

3.4 Analysis and Interpretation

As we have said before, our goal is to ascertain if any relationship exists between each of the proposed metrics (see section 2) and the maintenance time.

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 [20]. Therefore, we need others techniques, like machine learning techniques that allow us to build prediction models with two characteristics: they must be highly qualitative and more straightforward and intelligible to human beings.

In this work we a data analysis technique based on a method for induction of fuzzy rules [21,22]. 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 ease of understanding.

In Appendix A, we give an explanation of the method for induction of fuzzy rule systems we used for analysing the data obtained in the experiments.

Hereafter we show the results obtained by the application of the induction method we propose. We established a fuzzy rule system for the maintenance time. We used a learning set with X={set of our metrics} and Y={the values of the maintenance time obtained in our experiment}. We have obtained 207 values (9 class diagrams and 23 subjects) of this unknown function:

F(NC,NA,NM,NAssoc,NAgg,NDep,NGen,NAggH, NGenH,MaxHAgg,MaxDIT)=maintenance time

By applying the induction method (see Appendix A) to the data collected in the first experiment we have obtained the fuzzy rules shown in table 2, where MIN is the minimum value of the metrics, MAX is the maximum value of the metrics and [-] represents the whole domain of the metrics.

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:

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

Table 2. Maintenance time fuzzy model (results obtained in the first experiment done by "expert subjects")

NC	NA	NM	NGenH	Maintenance Time	ERROR	COV %
[20,23,MAX,MAX]	[33,75,MAX,MAX]	[66,95,MAX,MAX]	[1,2,MAX,MAX]	7.1364	1.8572	11.0850%
[20,23,MAX,MAX]	[33,75,MAX,MAX]	[66,95,MAX,MAX]	[MIN,MIN,1,2]	7.1364	0.0024	0.0140%
[20,23,MAX,MAX]	[-]	[MIN,MIN,31,65]	[-]	5.6948	0.0882	0.6799%
[20,23,MAX,MAX]	[MIN,MIN,33,75]	[66,95,MAX,MAX]	[-]	4.9968	0	0.0002%
[20,23,MAX,MAX]	[-]	[31,65,66,95]	[-]	4.8696	0.4950	11.7842%
[10,20,20,23]	[-]	[-]	[-]	4.7745	0.3806	11.0973%
[3,7,10,20]	[-]	[-]	[-]	3.0284	1.5579	53.3006%
[MIN,MIN,3,7]	[-]	[-]	[-]	2.8207	0.2947	11.8887%

Where: each row is on of the obtained rules; the columns **NC, NA, NM, NGenH** are the fuzzy sets associated with each metric name; the column Maintenance Time 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.

The rows represent the fuzzy rules. We can read rule 1 as ***"IF NC is in the fuzzy set [20,23,MAX,MAX] AND NA is in the fuzzy set [33,75,MAX,MAX] AND NM is in the fuzzy set [66,95,MAX,MAX] AND NGenH is in the fuzzy set [1,2,MAX,MAX] THEN Maintenance Time is 7.1364"*** . This rule makes 1.8572 error and has a data coverage percentage of 11.0850%.

Table 3 shows the set of fuzzy rules of the maintenance time model based on the data collected in the experiment performed by "novice subjects". This table could be read in the same manner as the previous table.

Table 3. Maintenance time fuzzy model (results obtained in the second experiment done by "novice subjects")

NC	NM	NAgg	NGen	NAggH	HAggMax	Maintenance time	ERROR	COV%
[-]	[-]	[1,3,MAX,MAX]	[5,14,MAX,MAX]	[1,2,MAX,MAX]	[-]	5.1513	1.1423	20.2120%
[-]	[-]	[1,3,MAX,MAX]	[5,14,MAX,MAX]	[0,1,1,2]	[-]	5.0721	0.0807	1.7702%
[-]	[-]	[1,3,MAX,MAX]	[MIN,MIN,0,1]	[0,1,1,2]	[-]	5.0423	0.1129	1.3296%
[-]	[-]	[1,3,MAX,MAX]	[MIN,MIN,5,14]	[1,2,MAX,MAX]	[-]	4.7090	1.1554	17.1221%
[-]	[-]	[MIN,MIN,1,3]	[-]	[1,2,MAX,MAX]	[-]	4.5583	0.2951	4.8853%
[3,7,8,9]	[7,21,31,65]	[1,3,MAX,MAX]	[0,1,1,3]	[0,1,1,2]	[1,2,2,3]	3.9781	2.0218	8.3748%
[3,7,8,9]	[MIN,MIN,7,21]	[1,3,MAX,MAX]	[0,1,1,3]	[0,1,1,2]	[1,2,2,3]	3.9781	0.0346	0.1435%
[8,9,10,17]	[-]	[1,3,MAX,MAX]	[0,1,1,3]	[0,1,1,2]	[1,2,2,3]	3.9780	0.2786	1.1540%
[3,7,8,9]	[31,65,MAX,MAX]	[1,3,MAX,MAX]	[0,1,1,3]	[0,1,1,2]	[1,2,2,3]	3.9780	0.0345	0.1428%
[MIN,MIN,3,7]	[-]	[1,3,MAX,MAX]	[0,1,1,3]	[0,1,1,2]	[1,2,2,3]	3.9710	0.0005	0.0020%
[-]	[-]	[1,3,MAX,MAX]	[1,3,5,14]	[0,1,1,2]	[-]	3.9226	0.0506	0.2558%
[-]	[-]	[1,3,MAX,MAX]	[0,1,1,3]	[0,1,1,2]	[MIN,MIN,1,2]	3.8872	0.0351	0.1891%
[10,17,MAX,MAX]	[-]	[1,3,MAX,MAX]	[0,1,1,3]	[0,1,1,2]	[1,2,2,3]	3.8170	0.0012	0.0082%
[-]	[-]	[1,3,MAX,MAX]	[0,1,1,3]	[0,1,1,2]	[2,3,MAX,MAX]	3.7247	0.0001	0.0008%
[-]	[-]	[MIN,MIN,1,3]	[-]	[0,1,1,2]	[-]	3.7087	0.5238	10.4783%
[-]	[-]	[-]	[-]	[MIN,MIN,0,1]	[-]	3.1515	0.5187	33.9205%

Where: each row is on of the obtained rules; the columns **NC, NM, Nagg, Ngen, NaggH, HaggMax** are the fuzzy sets associated with each metric name; the column Maintenance Time 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.

Now we will analyse the content of table 2 and 3 in order to corroborate our hypothesis "A close relation exists between the presented metrics and UML class diagram maintainability measured by the maintenance time". Making a general analysis we can point out some observations:

- Not all of the 11 metrics presented in section 2 are related with the maintenance time.
- In the first experiment only the metrics NC, NA, NM and NGenH show relation with the maintenance time, and, in the second the metrics NC, NM, NAgg, NGen, NAggH and HAggMax.
- The metrics NAssoc, NDep and MaxDIT do not seem to be related with the maintenance time. However this may be due to the design of the experiments, in which the value of those metrics did not take a great range of values.

However after making a thorough analysis we can see that the influence of each of the metrics (NC, NA, NM, NgenH, NAgg, NAggH, HAggMax) on each set of data (see table 2 and table 3) is not the same. If we consider that the rules which have a coverage percentage lower than 2% represent values with no significance, we can conclude that only 5 of the metrics determine the behaviour of the two set of data analysed. This fact could be explained thus:

- Data related to "expert subjects" (table 2): these data reflect a direct dependency on the metric NC and the maintenance time. Observing its values [MIN,MIN,3,7], [3,7,10,20] and [10,20,20,23] the maintenance time is 2.82 (1188% of the sample data), 3.02 (53.30 % of the sample data) and 4. 77 (11.09% of the sample data) respectively. This set of direct rules, taking into account the value of NC, cover 76.27% of the sample data. The remaining 23.73 % with the maximum value of NC [20, 23, MAX,MAX] is discriminated by a medium value of NM [31,65,66,95] with a maintenance time of 4.86

(11.78% of sample data) and a maximum value of NM [66,95,MAX,MAX] with the highest value of maintenance time being 7.13 (11.08% of the sample data). Therefore, in this case the metrics NC and NM are really relevant for the maintenance time.

- Data related to "novice subjects" (table 3): if the value of NAggH is low [MIN,MIN,0,1] the lowest value of the maintenance time obtained is 3.15 (33,9% of the sample data). In the case that NAggH has a medium value [0,1,1,2], it is necessary to observe the metric NAgg. A low value of NAgg [MIN,MIN,1,3] gives a maintenance time equal to 3.7 (10.47% of sample data), and a high value of NAgg [1,3,MAX,MAX] gives a maintenance time equal to 3.97 (8,37% of the sample data). With a high value of NAggH [1,2,MAX,MAX] and NAgg [MIN,MIN,1,3] the maintenance time is 4.55 (4.88%of the sample data), but if the value of NAgg is higher [1,3,MAX,MAX], the value of NGen discriminate the maintenance time. A small value of NGen [MIN, MIN, 5,14] produces a maintenance time equal to 4.7 (17,12%), and a higher value [5,14,MAX,MAX] produces a maintenance time equal to 5.15 (20,21% of the sample data). Therefore, in this case the metrics NAggH, NAggNC and NGen are really relevant for the maintenance time.

Despite these results, we are aware that it is necessary to have more data, taken from real projects in order to extract conclusive information, which will allow us to confirm if the presented metrics could really be used as early maintainability indicators, and could be used to predict UML class diagrams maintainability.

Moreover, the rules shown in table 2 and 3 can be used to draw other conclusions taking into account the expertise demonstrated by each group of subjects. If we observe the maintenance time of each table, the first corresponding to "expert subjects" (table 2) and the second corresponding to "novice subjects" we can note that the maximum and minimum maintenance times are different, [2.8,7.13] and [3.15,5.15] respectively.

The "novice subjects" have a higher minimum, that can be explained by the time that they spent in understanding more complex concepts that are reflected by the metrics NAggH, NAgg and NGen, which are related to aggregation and generalisation relationships. Unlike the "expert subjects" who quickly focus their attention on more global concepts such as the classes and methods reflected in the metrics NC and NM. This means that in the more straightforward examples the "expert subjects" take less time, and in the more complex examples they take longer as they did not focus their attention right from the beginning on the more complex structures and relationships.

In addition to all that has been discussed so far, we also observed that the aggregation relationships did not appear to affect the "expert subjects" which could be explained by the fact that they are used to working with other modelling techniques such as the entity relationship model. The number of classes (NC) and the number of methods (NM) affect the "expert subjects" and "novice subjects".

3.5 Validity evaluation

We will discuss the various issues that threat to the validity of the empirical study and the way we attempted to alleviate them:

- **Threats to Construct Validity.** The dependent variable we used is the maintenance time, i.e., the time each subject spent performing the tasks related to the modifications arising from the new requirements, so we consider this variable constructively valid. For construct validity of the independent variables, we have to address the question to what 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 [23]. 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.
- **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 [17], 1999), in software engineering experiments when dealing with small samples, variations in participant skills are a major concern that is difficult to address fully by randomisation or blocking. In the first experiment, professors and students had the same degree of experience in modelling with UML. The same occurred with the subjects in the second experiment.
  KNOWLEDGE OF THE UNIVERSE OF DISCOURSE AMONG CLASS DIAGRAMS. The class diagrams were from different universes of discourse but they were general enough to be easily understood by each of the subjects, and also a brief specification of them was added to each diagram. In this way, the knowledge of the domain does not affect internal validity.
  ACCURACY OF SUBJECT RESPONSES. Subjects assumed the responsibility of performing the maintenance tasks and writing down the maintenance

time. The "expert subjects" have approximately three years of experience in OO software design and implementation, and the "novice subjects" have approximately one year's experience. Therefore, we think their responses can be considered valid. However we are aware that not all of them have exactly the same degree of experience, and the subjects that spend less time may have more experience .

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 the tests appeared.

FATIGUE EFFECTS. On average the experiment lasted for less than one hour, so fatigue was not very relevant. Also, the different order of 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 the first experiment have participated voluntarily, in order to help us in our research. We motivated students to participate in the experiments, by 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 opportunity to do the experiment.

OTHER FACTORS. Plagiarism and influence between students really could not be controlled. Students were told that talking to each other was forbidden, but they did the experiments alone without any control, so we had to trust them as far as that was concerned.

Seeing the results of the experiments (section 3.4) we can conclude that empirical evidence of a relationship existing between the independent and the dependent variables exists. However 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 causality really be demonstrated.

- **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 threats to 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 [14].

4. Prediction of class diagram maintainability

The fuzzy rules system shown in table 2 and table 3 could be used as prediction models for the maintenance time. By means of an example we will demonstrate how the maintenance time of a class diagram can be predicted using the model presented in table 3. Suppose that we want to answer the following question, **"What is the value of the Maintenance Time, if we know that NC is 20, NM is 65, NAgg is 3, NGen is 7, NAggH is 2 and HAggMax is 1?"**. This question, can be answered by inference following table 4.

Table 4. Inference of a new value of the maintenance time

NC=20	NM=65	Nagg=3	NGen=7	NAggH=2	HAggMax=1	Maintenance Time	MIN	MIN*Maintenace Time
1	1	1	0.23	1	1	**5.151**	0.23	1.145
1	1	1	0.23	0	1	**5.072**	0	0.000
1	1	1	0	0	1	**5.042**	0	0.000
1	1	1	0.78	1	1	**4.709**	0.78	3.663
1	1	0	1	1	1	**4.558**	0	0.000
0	0	1	0	0	0	**3.978**	0	0.000
0	0	1	0	0	0	**3.978**	0	0.000
1	1	1	0	0	0	**3.978**	0	0.000
0	1	1	0	0	0	**3.978**	0	0.000
0	1	1	0	0	0	**3.971**	0	0.000
1	1	1	0.78	0	1	**3.923**	0	0.000
1	1	1	0	0	1	**3.887**	0	0.000

0	1	1	0	0	0	**3.817**	0	0.000
1	1	1	0	0	0	**3.725**	0	0.000
1	1	0	1	0	1	**3.709**	0	0.000
1	1	1	1	0	1	**3.152**	0	0.000
					SUM		**1**	**4.807**

Where: the columns **NC, NM, NAgg, NGen, NAggH, HAggMax** represent the degree of membership of each metric to the fuzzy set; the column **Maintenance Time** represents the output or the consequent of the rules; the column **MIN** represents the minimum of the degree of membership of the antecedents and the column **MIN*Maintenance Time** represents the contribution of each rule to the final solution 4.807 minutes.

Using approximate reasoning [24] and with a simplified consequent, detailed in table 3, we obtain the value of the maintenance time as 4.807/1=4.807minutes. This value is obtained by making a weighted average of the output (maintenance time) of each rule with its degree of membership. Even though this is a very simple example, it illustrates that it is possible to predict the class diagram maintenance time from the metrics values, early in the development of OOIS. These predictions, applied to real projects, can help OOIS designers take better decisions.

5. Conclusions and future work

In this paper, we have presented a set of measures for evaluating UML class diagram structural complexity. These metrics allow OO designers:

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

From structural complexity metrics we have built a prediction model for class diagram maintainability using a method for induction of fuzzy rules. The data used to build those prediction models was collected through two controlled experiments. The first experiment was carried out by expert OOIS designers and the second (a replication of the first) was carried out by novice OOIS designers.

These experiments reveal that not all of the metrics are related with class diagram maintainability. Only NC, NM, NAgg, NGen and NAggH show a close relationship, which indicate that they could be used to predict class diagram maintainability, early in the OOIS life-cycle. Nevertheless, given that these conclusions can only be considered as preliminary we cannot discard the other metrics. We are aware that we need to do more metric validation, especially taking data from "real cases" to derive some design heuristics that could/should be included in design tools. This could really help IS designers take better decisions in their design tasks, which is the most important goal that any measurement proposal must pursue if it aims to be useful [25].

We will also focus our research on measuring other quality factors like those proposed in the ISO 9126 [5], 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 [23,26]. This is an area which needs further investigation [28].

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).

References

[1] B. Henderson-Sellers, *Object-Oriented Metrics - Measures of complexity*, Prentice-Hall, Upper Saddle River, New Jersey, 1996.

[2] A. Melton (ed.), *Software Measurement*, International Thomson Computer Press, London, 1996.

[3] N. Fenton, S. Pfleeger, *Software Metrics: A Rigorous Approach*, 2nd. Edition, Chapman & Hall, London, 1997.

[4] H. Zuse, *A Framework of Software Measurement*, Walter de Gruyter, Berlin, 1998.

[5] ISO/IEC 9126-1.2. Information technology- Software product quality – Part 1: Quality model, 1999.

[6] S. Chidamber, and C. Kemerer, "A Metrics Suite for Object Oriented Design", *IEEE Transactions on Software Engineering*, 20(6), 1994, pp. 476-493.

[7] M. Lorenz, and J. Kidd, *Object-Oriented Software Metrics: A Practical Guide*, Prentice Hall, Englewood Cliffs, New Jersey, 1994.

[8] F. Brito e Abreu, and R. Carapuça, "Object-Oriented Software Engineering: Measuring and controlling the development process", *4th Int Conference on Software Quality*, Mc Lean, Va, USA, 1994.

[9] M. Marchesi, "OOA Metrics for the Unified Modeling Language", *Proceedings of the 2nd Euromicro Conference on Software Maintenance and Reengineering*, 1998, pp. 67-73.

[10] M. Genero, M. Piattini, and C. Calero, "Early Measures For UML class diagrams", *L'Objet*, 6(4), Hermes Science Publications, 2000, pp. 489-515.

[11] Object Management Group, UML Revision Task Force, OMG Unified Modeling Language Specification, v. 1.3. document ad/99-06-08, 1999.
[12] N. Schneidewind, "Methodology For Validating Software Metrics", *IEEE Transactions of Software Engineering*, 18(5), 1992, 410-422.
[13] B. Kitchenham, S. Pflegger, and N. Fenton, "Towards a Framework for Software Measurement Validation", *IEEE Transactions of Software Engineering*, 21(12), 1995, pp. 929-943.
[14] V. Basili, F. Shull, and F. Lanubile, "Building knowledge through families of experiments", *IEEE Transactions on Software Engineering*, 25(4), 1999, pp. 435-437.
[15] C. Wohlin, P. Runeson, M. Höst, M. Ohlson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction*, Kluwer Academic Publishers, 2000.
[16] D. Perry, A. Porte, and L. Votta, "Empirical Studies of Software Engineering: A Roadmap", *Future of Software Engineering*, Ed. Anthony Finkelstein, ACM, 2000, pp. 345-355.
[17] L. Briand, S. Arisholm, F. Counsell, F. Houdek, and P. Thévenod-Fosse, "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.
[18] V. Basili, and D. Weiss, "A Methodology for Collecting Valid Software Engineering Data", *IEEE Transactions on Software Engineering*, 10, 1984, pp. 728-738.
[19] V. Basili, and H. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments", *IEEE Transactions on Software Engineering*, 14, 1988, pp. 758-773.
[20] S. Morasca, and G. Ruhe, Guest Editors "Introduction: Knowledge Discovery From Empirical Software Engineering Data", *International Journal of Software Engineering and Knowledge Engineering*, 9 (5), 1999, pp. 495-498.
[21] L. Zadeh, "Fuzzy sets", *Information and control*, 1965, pp. 338-353.
[22] L. Zadeh, "The Concept of Linguistic Variable and its Applications to Approximate Reasoning Part I", *Information Sciences*, 8, 1973, pp. 199-249.
[23] M. Sugeno, "An Introductory Survey of Fuzzy Control", *Information Sciences*, 36, 1985, pp. 59-83.
[24] N. Fenton, and M. Neil, "Software Metrics: a Roadmap", *Future of Software Engineering*, Ed. Anthony Finkelstein, ACM, 2000, pp. 359-370.
[25] G. Poels, "On the Measurement of Event-Based Object-Oriented Conceptual Models", *4th International ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering*, Cannes, France, 2000.
[26] G. Poels, and G. Dedene, "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, 2000, pp. 499-512.
[27] F. Brito e Abreu, H. Zuse, H. Sahraoui, and W. Melo, "Quantitative Approaches in Object-Oriented Software Engineering". *Object-Oriented technology: ECOOP'99 Workshop Reader*, Lecture Notes in Computer Science 1743, Springer-Verlag, 1999, pp. 326-337.
[28] L. Breiman, J. Friedman, R. Olshen, and C. Stone, *Classification and Regression Trees*, Wadsworth, Belmont, CA., 1984.
[29] L. Linares, M. Delgado, and A. Skarmeta, "Regression by fuzzy knowledge bases", *Proceedings of the 4th European Congress on Intelligent Techniques and Soft Computing*, Aachen, Germany, 1996, pp. 1170-1176.
[30] M. Delgado, A. Gómez Skarmeta, and L. Jiménez, "Induction of fuzzy rules", *International Journal of Intelligent Systems*, 16, 2001 (to appear).

Appendix A

In order to explain the method for induction of a fuzzy rule system, first we make some introductory comments.

Let $S=\{s_1, \ldots, s^n\}$ be a set of data, which are defined by the value given by a set of variables $X=\{X^1, \ldots, X^d, Y\}$, $s_i=(x^1_i, \ldots, x^d_i, y_i)$. Let's suppose there is a function F which is only known at the points of S so that $F(s_i)=y_i$. The objective of regression, which is defined classically as a parametric function F', is to minimise the distance between the sample output values y_i and the predicted value $F'(s_i)$.

The regression problem differs from the classification problem in that the output variable can be continuous, where as in classification this is strictly categorical. From this perspective, classification can be thought to be a subcategory of regression. Recursive partition methods for classification problems such as ID3 decision tree have been applied, as a regression method by restrictions, in the CART program [28], developed in the statistical research community.

The program CART (classification and regression tree) is based on the building of a tree structure where the regions are defined by possible answers to a set of questions raised about the variables that define the problem. This approach creates a set of disjoint regions $SR=\{r_1, \ldots, r_p\}$ of the global domain of the problem. These partitions are obtained according to the kind of questions and answers that we have formulated. Our method generalises the kind of partitions obtained, using the method proposed in [29] and [30], based on the fuzzy set theory [21]. In the fuzzy set theory the membership of an element x (which belongs to a domain X) to a fuzzy set A, is in the interval [0,1]. If A(x) is the membership function, A(x)=0 means that x is not a member of A, and A(x)=1 means that x is a full member of A. A(x) can also take other values between the interval [0,1], which graduate the membership of x to the fuzzy set A. We follow Linares et al.'s idea [29] in order to define a fuzzy classification and regression tree (FCART), which produce a fuzzy rule system that represents our prediction model.

Hereafter, we will explain how to obtain the rules. Let A_T be a fuzzy set defined over the set S in the node T, such as $A_T: S \rightarrow [0,1]$. In the root node this fuzzy set is $A_{root}: S \rightarrow 1$. We define the output value at node T by the

membership value of each point s_i, $A_T(s_i)$ and the output value in this point y_i.

$$F''(T) = \frac{\sum_{i=1}^{n} A_T(s_i)^m * y_i}{\sum_{i=1}^{n} A_T(s_i)^m}$$

Where $A_T(x)$ is defined by $\min_{j=1}^{d} A_T^j(x^j)$, and $A_T^j(x^j)$ is a membership of a fuzzy set defined over j-metric domain. The value of m is a real number greater than 1 which weights the contribution of the membership value.

The estimated error is defined by:

$$Err(T) = \frac{\sum_{i=1}^{n} (F''(T) - y_i)^2 * A_T(s_i)^m}{\sum_{i=1}^{n} A_T(s_i)^m}$$

Now, our problem is how to create the set of questions to split the node T. The questions were formulated for each variable, obtaining a binary fuzzy partition for each one. We supposed a binary fuzzy partition for the fuzzy set of node T by the fuzzy set A^j_T of variable j, this is $p^j_T = \{B(x), C(x)\}$, when $A^j_T(x)=B(x)+C(x)$. This partition originates two new nodes and two new fuzzy sets for them:

$$A_{T_1}(s_i) = min(A_T(s_i), B(x_i^j))$$

$$A_{T_2}(s_i) = min(A_T(s_i), C(x_i^j))$$

Following the CART program, we defined the proportion for B(x) and C(x) in relation to fuzzy set A^j_T as:

$$P(T_1) = \frac{\sum_{i=1}^{n} B(x_i^j)}{\sum_{i=1}^{n} A_T^j(x_i^j)}$$

The quality of the partition can be estimated through the equation:

$$C(T, p^j) = Err(T_1) * P(T_1) + Err(T_2) * P(T_2)$$

where p^j is the fuzzy partition of the variable j.

We will select the p^* which minimises the value of $C(T, p^j)$. This approach to creating the questions gives rise to a hierarchical fuzzy partition for each variable. This split process raises relevant metrics to define the model. The partition process stops when a stop criterion is raised. In this case we look for larger estimated error. So that stop criterion can be:

$$ERROR = \max_{T \in \overline{T}} Err(T) \leq \varepsilon$$

Where $\overline{T}$ is the set of leaf nodes of the tree, and each one represents a region for our solution. The output value y_i' for an input value s_i is:

$$F'(s) = \frac{\sum_{T \in \overline{T}} A_T(s)^m * F''(T)}{\sum_{T \in \overline{T}} A_T(s)^m}$$