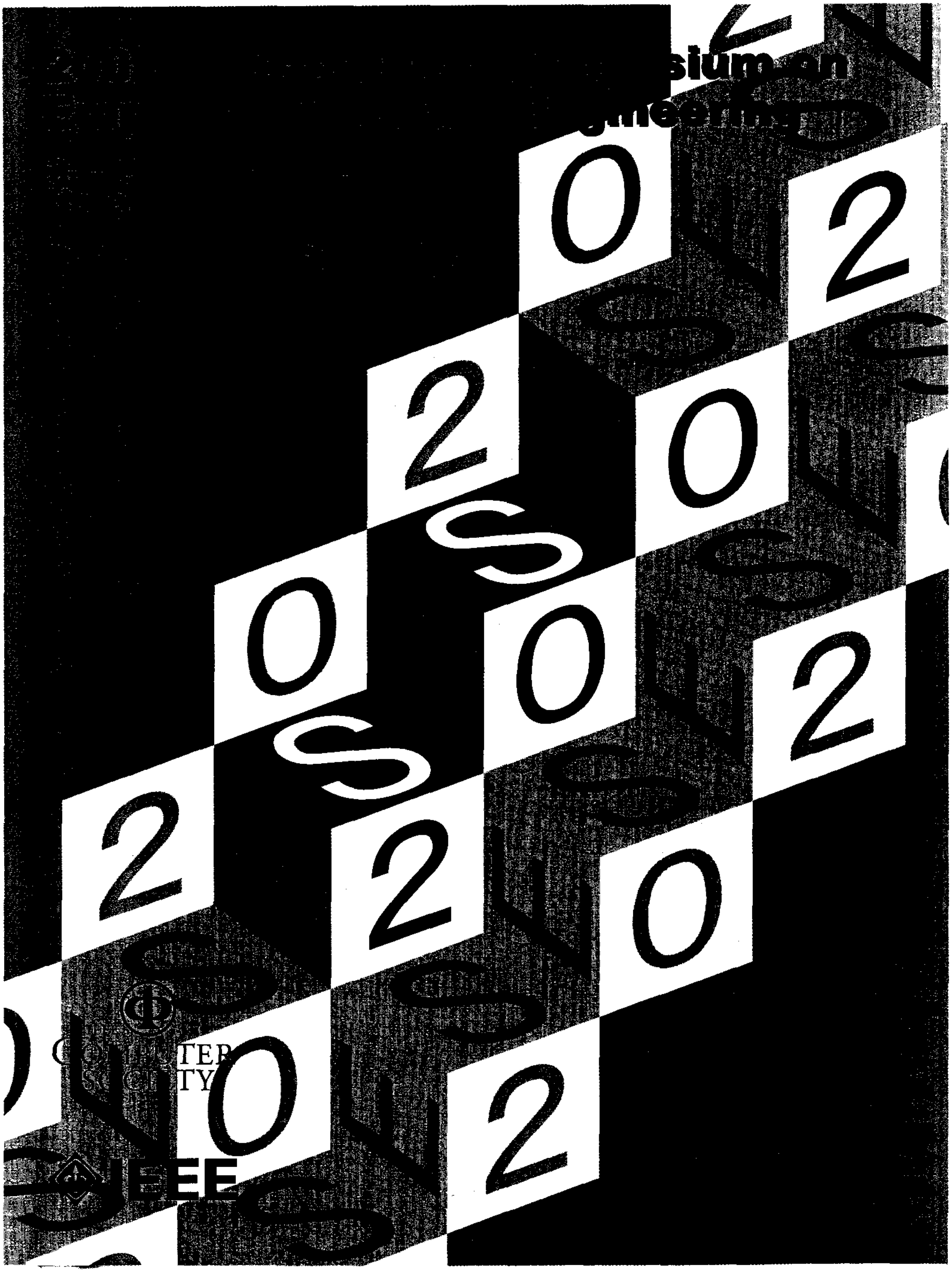

sium on

gineering

COMPUTER SOCIETY

IEEE

ISESE 2002

Published by the IEEE Computer Society
10662 Los Vaqueros Circle
P.O. Box 3014
Los Alamitos, CA 90720-1314

IEEE Computer Society Order Number PR01796
Library of Congress Number 2002109465
ISBN 0-7695-1796-X

ISBN 0-7695-1796-X
90000
9 780769 517964

PROCEEDINGS

Table of Contents

Keynote Address

Session 1: Experimental Methods

Session 2A: Software Quality

Session 2B: Estimation

Session 3: Process Improvement

Session 4: Inspections

Session 5: Testing

Session 6: Architecture and Design

Empirical Validation of Class Diagram Metrics

Marcela Genero, Mario Piattini and Coral Calero
Alarcos Research Group
Department of Computer Science, University of Castilla-La Mancha.
Paseo de la Universidad, 4, 13071, Ciudad Real (Spain)
{Marcela.Genero, Mario.Piattini, Coral.Calero}@uclm.es

Abstract

As a key early artefact in the development of OO software, the quality of class diagrams is crucial for all later design work and could be a major determinant for the quality of the software product that is finally delivered. Quantitative measurement instruments are useful to assess class diagram quality in an objective way, thus avoiding bias in the quality evaluation process. This paper presents a set of metrics -based on UML relationships- which measure UML class diagram structural complexity following the idea that it is related to the maintainability of such diagrams. Also summarized are two controlled experiments carried out in order to gather empirical evidence in this sense.

As a result of all the experimental work, we can conclude that most of the metrics we proposed (NAssoc, NAgg, NaggH, MaxHAgg, NGen, NgenH and MaxDIT) are good indicators of class diagram maintainability. We cannot, however, draw such firm conclusions regarding the NDep metric.

Keywords. OO high-level metrics, structural complexity, class diagrams, maintainability, UML, empirical validation

1. Introduction

In the development of OO software, the class diagram is a key early artefact that lays the foundation of all later design and implementation work. The early focus on class diagram quality may help software designers build better OO software, without unnecessary revisions at later development stages when changes are more expensive and more difficult to perform. It is in this arena where software measurement plays an important role, because the early availability of metrics contributes to class diagram quality evaluation in an objective way avoiding bias in the quality evaluation process. Moreover, metrics provide a valuable and objective insight into specific ways of enhancing each of the software quality characteristics.

Given that maintenance is (and will continue to be) the major resource consumer of the whole software life cycle, maintainability has become one of the software product quality characteristics that software development organisations are more concerned about. However, we are aware that maintainability is an external quality attribute that can only be measured when the OO software product is (nearly) finished. Therefore, it is necessary to have early indicators of such qualities based, for example, on the structural properties of class diagrams [1].

As most of the existing OO measures [2],[3],[4], [5] are related to measures applied to code or to detailed design, they provide information too late to lead us in building maintainable OO software. So, after a thorough review of some of the existing OO measures, applicable to class diagrams at high-level design stage [6],[7],[8][9] we have proposed [10],[11] a set of UML class diagram structural complexity measures based on the use of UML relationships (associations, generalizations, aggregations and dependencies), see table 1, where also, traditional metrics such as Number of Classes, Number of Methods and Number of Attributes are included. These metrics presented have been developed in a methodological way which consists of three main steps: metric definition, and theoretical and empirical validation [12]. Even though the three steps are relevant, in this paper we shall only deal with the empirical validation. More details on the definition and theoretical validation can be found in [11].

As the proposal of metrics is of no value if their practical use is not demonstrated empirically [13],[2],[14],[15],[16] either by means of case studies taken from real projects or by controlled experiments, our main motivation is to investigate, through experimentation, if the metrics we proposed for UML class diagram structural complexity (internal quality attribute) are related to some class diagram maintainability (external quality attribute) sub-characteristics: understandability, analyzability and modifiability [17][1]. If such a relationship exists and is confirmed by empirical studies, we will have really

[1] Even though understandability has not been considered as a maintainability sub-characteristic by the ISO 9126 [17] we include it because several works related to software measurement consider understandability to be a factor that influences maintainability [2],[18],[19].

0-7695-1796-X/02 $17.00 © 2002 IEEE

obtained early indicators of class diagram maintainability. These indicators will allow OO software designers to take better decisions early in the OO software development life cycle, thus contributing to the development of better quality OO software.

Table 1. Metrics for UML class diagram structural complexity

Metric name	Metric definition
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 AGGREGATIONS HIERARCHIES (NAggH)	The total number of aggregation hierarchies (whole-part structures) within a class diagram.
NUMBER OF GENERALISATIONS HIERARCHIES (NGenH)	The total number of generalisation hierarchies within a class diagram.
MAXIMUM DIT (MaxDIT)	It is the maximum of the DIT (Depth of Inheritance Tree) values 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 (MaxHAgg)	It is the maximum of the HAgg values 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.
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

This paper is organized as follows: In sections 2 and 3 we summarize how we carried out the two controlled experiments for empirically validating the metrics. The comparison of the results obtained in both experiments comes in section 4, and finally and in section 5 some conclusions are presented together with future work.

2. First Experiment

In this section we describe the first experiment we have carried out to empirically validate the proposed measures as early maintainability indicators. We have followed some suggestions provided by Wohlin et al. [20], Perry et al. [21] and Briand et al. [1] on how to perform controlled experiments. To describe the experiment we use (with only minor changes) the format proposed by Wohlin et al. [20] comprising the following main tasks: definition, planning, operation, analysis and interpretation, validity evaluation and presentation and package.

2.1 Definition

Using the GQM template [22] for goal definition, the goal of the experiment is defined as follows:

Analyse *UML class diagram structural complexity metrics*
For the purpose of *Evaluating*
With respect to *the capability to be used as class diagram maintainability indicators*
From the point of view of *OO Software designers*
In the context of *Undergraduate Computer Science students and professors of the Software Engineering area at the Department of Computer Science at the University of Castilla-La Mancha*

2.2. Planning

2.2.1. Context selection. The context of the experiment is a group of undergraduate students and professors of the Software Engineering area, and hence the experiment is run off-line (not in an industrial software development environment). The subjects were seven professors and ten students enrolled in the final-year of Computer Science at 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, i.e., what indicators can be used for the maintainability of class diagrams? With this end in view it investigates the correlation between class diagram structural complexity metrics and maintainability sub-characteristics.

2.2.2. Selection of subjects. The subjects are chosen for convenience, i.e., the subjects are undergraduate students and professors who have experience in the design and development of OOIS using UML.

2.2.3. Variable selection. The independent variable is the class diagram structural complexity. The dependent variables are three maintainability sub-characteristics: understandability, analyzability and modifiability.

2.2.4. Instrumentation. The objects were UML class diagrams. The independent variable was measured through the metrics we proposed. The dependent variables were measured according to the subject's ratings.

2.2.5. Hypothesis formulation. We wish to test the following hypotheses:

- Null hypothesis, H_0: There is no significant correlation between the structural complexity metrics (NC, NA, NM, NAssoc, NAgg, NDep, NGen, NAggH, NGenH, MaxHAgg, MaxDIT) and the subject's rating of three maintainability sub-characteristics, such as understandability, analyzability and modifiability.
- Alternative hypothesis, H_1: There is a significant correlation between the structural complexity metrics (NC, NA, NM, NAssoc, NAgg, NDep, NGen, NAggH, NGenH, MaxHAgg, MaxDIT) and the subject's rating of three maintainability sub-characteristics, such as understandability, analyzability and modifiability.

2.2.6. Experiment design. We selected a within-subject design experiment, i.e., all the tests (experimental tasks) had to be solved by each of the subjects. The tests were put in a different order for each subject.

2.3 Operation

2.3.1. Preparation. By the time the experiment was done all of the students had taken two courses in Software Engineering, in which they learnt in depth how to design OO software using UML. All the selected professors had more than four years of experience in the design and development of OO software using UML. Moreover, subjects were given an intensive training session before the experiment took place. However, 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 handed to the subjects, consisting of twenty eight UML class diagrams of the same universe of discourse, related to Bank Information Systems. The structural complexity of each diagram is different, because as table 2 shows, the values of the metrics are different for each diagram.

Table 2. Metric values for each class diagram

	NC	NA	NM	NAssoc	NAgg	Ndep	NGen	NAggH	NGenH	MaxHAgg	MaxDIT
D0	2	4	8	1	0	0	0	0	0	0	0
D1	3	6	12	1	1	0	0	1	0	1	0
D2	4	9	15	1	2	0	0	1	0	2	0
D3	3	7	12	3	0	0	0	0	0	0	0
D4	5	14	21	1	3	0	0	2	0	2	0
D5	3	6	12	2	0	0	0	0	0	0	0
D6	4	8	12	3	0	1	0	0	0	0	0
D7	6	10	14	2	2	0	2	1	1	2	1
D8	3	9	12	1	0	1	0	0	0	0	0
D9	7	14	20	2	3	0	2	1	1	2	1
D10	9	18	26	2	3	0	4	1	2	3	1
D11	7	18	37	3	3	0	2	1	1	3	1
D12	8	22	35	3	2	1	2	1	1	2	1
D13	5	9	26	0	0	0	4	0	1	0	2
D14	8	12	30	0	0	0	10	0	1	0	3
D15	11	17	38	0	0	0	18	0	1	0	4

D16	20	42	76	10	6	2	10	2	3	2	2
D17	23	41	88	10	6	2	16	2	3	4	3
D18	21	45	94	6	6	1	20	2	2	4	4
D19	29	56	98	12	7	3	24	3	4	4	4
D20	9	28	47	1	5	0	2	2	1	4	1
D21	18	30	65	3	5	0	19	1	2	3	4
D22	26	44	79	11	6	0	21	2	5	4	3
D23	17	32	69	1	5	0	19	1	1	2	5
D24	23	50	73	9	7	2	11	3	4	4	1
D25	22	42	84	14	4	4	16	2	3	2	3
D26	14	34	77	4	9	0	7	2	2	3	4
D27	17	34	47	6	6	0	11	3	2	2	2

Each diagram had a test enclosed which includes the description of three maintainability sub-characteristics: understandability, analyzability and modifiability. Each subject had to rate each sub-characteristic using a scale consisting of seven linguistic labels. For example for understandability we proposed the following linguistic labels shown in table 3.

Table 3. Linguistic labels for understandability

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 chose seven linguistic labels because we considered they are enough to cover all the possible categories of our variable, the understandability. The selection of and odd number of labels has been done based on some suggestions provided in [23],[24] where they justify that and odd number contribute to obtain better results due to they are balanced.

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.

2.3.2. Execution. The subjects were given all the materials described in the previous paragraph. We explained to them how to carry out the tests. 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 subjects' rating obtained from the responses of the experiment and the metric values automatically calculated by means of a metric tool we had designed.

2.3.3. Data validation. We collected all the tests, checking if they were complete. As all of them were complete and the subjects had at least medium experience in building class diagrams (this fact was corroborated analysing the responses of the debriefing questionnaire) we consider their subjective evaluation reliable.

2.4. Analysis and interpretation

First we summarised the data collected. We had the metric values calculated for each class diagram, and we calculated the median of the subjects' rating for each maintainability sub-characteristic. So this is the data we want to analyse to test the hypotheses stated above.

We applied the Kolmogrov-Smirnov test to ascertain if the distribution of the data collected was normal or not. As the data were non-normal we decided to use a non-parametric test like Spearman's correlation coefficient, with a level of significance $\alpha = 0.05$, which means the level of confidence is 95%.

Using Spearman's correlation coefficient, each of the metrics was correlated separately to the median of the subject's rating of understandability, analyzability and modifiability (see table 4).

Table 4. Spearman's correlation between the metrics and understandability, analysability and modifiability

	NC	NA	NM	NAssoc	NAgg	NDep	NGen	NAggH	NGenH	MaxHAgg	MaxDIT
Understandability	0.912 p=0	0.892 p=0	0.859 p=0	0.775 p=0	0.789 p=0	0.554 p=0.002	0.833 p=0	0.683 p=0	0.857 p=0	0.718 p=0	0.677 p=0
Analysability	0.926 p=0	0.896 p=0	0.883 p=0	0.724 p=0	0.812 p=0	0.529 p=0.04	0.848 p=0	0.693 p=0	0.863 p=0	0.684 p=0	0.759 p=0
Modifiability	0.943 p=0	0.907 p=0	0.909 p=0	0.730 p=0	0.788 p=0	0.525 p=0.04	0.881 p=0	0.676 p=0	0.891 p=0	0.673 p=0	0.805 p=0

For a sample size of 28 (median values for each diagram) and $\alpha = 0.05$, the Spearman cutoff for accepting H_0 is 0.48 [25],[26]. Because the computed Spearman's correlation coefficients (see table 4) are above the cutoff, and the p-value < 0,01, the null hypothesis H_0, is rejected. Hence, we can conclude that there is a significant correlation between the UML class diagram structural complexity metrics and subject's rating of understandability, analysability and modifiability.

2.5. Validity evaluation

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

2.5.1. Threats to conclusion validity. The conclusion validity defines the extent to which conclusions are statistically valid. The only issue that could affect the statistical validity of this study is the size of the sample data (476 values, 28 diagrams and 17 subjects), which is perhaps not enough for both parametric and non-parametric statistic tests [26]. We are aware of this, so we will consider the results of the experiment only as preliminary findings.

2.5.2. Threats to construct validity. The construct validity is the degree to which the independent and the dependent variables are accurately measured by the measurement instruments used in the study. The dependent variables are three maintainability sub-characteristics: understandability, analysability and modifiability. We proposed subjective metrics for them (using linguistic variables), based on the judgement of the subjects. As the subjects involved in this experiment have medium experience in OOIS design using UML we think their ratings could be considered significant. The construct validity of the metrics used for the independent variables is guaranteed by Poels and Dedene's framework[2] [27] used to define and validate them.

2.5.3. Threats to Internal Validity. The internal validity is the degree to which conclusions can be drawn about cause - effect of independent variables on the dependent variables. 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. [18] remarks when dealing with small samples in software engineering experiments, 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 approximately the same degree of experience in modelling with UML[3].
- Knowledge of the universe of discourse among class diagrams. Class diagrams were from the same universe of discourse, the only variant being the number of attributes, classes or associations, i.e., their constituent parts. Consequently, knowledge of the domain does not affect the internal validity.
- Accuracy of subject responses. Subjects assumed the responsibility for rating each maintainability sub-characteristic. As they have medium 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. The subjects were given the test in a different order, to cancel out learning effects. Subjects were required to answer in the order in which the tests appeared.
- Fatigue effects. On average the experiment lasted for less than one hour (this fact was corroborated summing the total time for each subject), so fatigue was not very relevant. Also, the different order in the tests helped to cancel out these effects.
- Persistence effects. In order to avoid persistence effects, the experiment was run with subjects who had never done a similar experiment.
- Subject motivation. All the professors who were involved in this experiment have participated

[2] This framework is based on Measurement Theory [28] and not also assure that the metrics measure the attribute they purpose to measure but also that the metrics are characterised in the ratio scale.

[3] We argue this because the students are Ph.D. students and students of the final-year, and professors are young professors, who has been graduated one or two years ago.

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.

- Other factors. Plagiarism and influence among students could not really be controlled. Students were told that talking with each other was forbidden, but they did the experiment alone without any supervision, so we had to trust them as far as that was concerned. We are conscious that this aspect at some extent could can threat to the validity of the experiment, but in that moment it was impossible to join all the subjects together. We are planning to replicate this experiment in a more controlled environment.

2.5.3. Threats to external validity. The external validity is the degree to which the results of the research can be generalised to the population under study and other research settings. The greater the external validity, the more the results of an empirical study can be generalised to actual software engineering practice. Two threats of validity have been identified which limit the possibility of applying any such generalisation:

- Materials and tasks used. In the experiment we tried to use class diagrams which can be representative of real cases. Related to the tasks, the judgement of the subjects is to some extent subjective, and does not represent a real task. So 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 [13].

2.6. Presentation and package

As the diffusion of the experimental data is important to external replication [29] of the experiments we have put all the material of this experiment on our web site http:\\alarcos.inf-cr.uclm.es.

3. Second experiment

As the majority of the steps are identical to those of the first experiment we will only point out those issues which are different. The subjects were ten professors and twenty students enrolled in the final-year of Computer Science at 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 dependent variable was measured by the time the subjects spent carrying out the tasks required in the experiment. We called this time "maintenance time". Maintenance time comprises the time to comprehend the class diagram, to analyse the required changes and to implement them. Our assumption here is that, for the same modification task, the faster a class diagram can be modified, the easier it is to maintain.

We wish to test the following hypotheses:

- Null hypothesis, H0: There is no significant correlation between structural complexity metrics (NC, NA, NM, NAssoc, NAgg, NDep, NGen, NAggH, NGenH, MaxHAgg, MaxDIT) and maintenance time.
- Alternative hypothesis, H1 : There is a significant correlation between structural complexity metrics (NC, NA, NM, NAssoc, NAgg, NDep, NGen, NAggH, NGenH, MaxHAgg, MaxDIT) and maintenance time.

The material we gave to the subjects consisted of a guide explaining UML notation and nine UML class diagrams of different application domains, that were easy enough to be understood by each of the subjects. The diagrams have different structural complexity, covering a broad range of metric values (see table 5).

Table 5. Metric values for each class diagram.

	NC	NA	NM	NAssoc	NAgg	NDep	NGen	NAggH	NGenH	MaxDIT	MaxHAgg
D1	7	11	22	1	0	0	5	0	1	2	0
D2	8	12	31	1	6	0	1	1	1	1	2
D3	3	17	24	2	0	0	0	0	0	0	0
D4	10	12	21	15	3	0	0	2	0	0	1
D5	9	19	29	3	3	0	3	3	1	2	1
D6	7	16	7	6	0	0	0	0	0	0	0
D7	23	33	66	4	5	2	16	2	3	3	3

D8	20	30	65	6	5	0	14	4	3	3	2
D9	23	65	80	20	3	2	3	3	1	2	3

Each diagram had an enclosed test that included a brief description of what the diagram represented and two new requirements for the class diagram. Each subject had to modify the class diagrams according to the new requirements and to specify the start and end time. The difference between the two is what we call maintenance time (expressed in minutes and seconds). The modifications to each class diagram were similar, including adding attributes, methods, classes, etc.

We collected all the data including the modified class diagrams with the maintenance time obtained from the responses of the tests and the metrics values automatically calculated by means of a metric tool we designed.

Once the data was collected, we controlled if the tests were complete and if the modifications had been done correctly. We discarded the tests of seven subjects, which included a required modification that was done incorrectly. Therefore, we took into account the responses of 23 subjects.

We had the metric values calculated for each class diagram (see table 5), and we calculated the mean of the maintenance time. So this is the data we want to analyse to test the hypotheses stated above. We applied the Kolmogrov-Smirnov test to ascertain if the distribution of the data collected was normal. As the data were non-normal we decided to use a non-parametric test like Spearman's correlation coefficient, with a level of significance $\alpha = 0.05$, correlating each of the metrics separately with maintenance time (see table 6).

Table 6. Spearman´s correlation coefficients between metrics and maintenance time

	NC	NA	NM	NAssoc	NAgg	NDep	NGen	NAggH	NGenH	Max HAgg	Max DIT
Maintenance Time	0.941 p=0	0.803 p=0.009	0.795 p=0.01	0.671 p=0.006	0.667 p=0.049	0.411 p=0.272	0.728 p=0.04	0.759 p=0.018	0.719 p=0.029	0.840 p=0.005	0.669 p=0.04

For a sample size of 9 (mean values for each diagram) and $\alpha = 0.05$, the Spearman cutoff for accepting H_0 is 0.66 [25],[26]. Because the computed Spearman's correlation coefficients (see table 6) for all the metrics, except for NDep, are above the cutoff, and the p-value < 0,05, the null hypothesis H_0, is rejected . Hence, we can conclude that there is a significant correlation between all the metrics (except Ndep) and the maintenance time.

So, NDep is the only one that has a no correlation, but this could be explained by the fact that in most of the selected diagrams NDep took the value 0 (see table 5). So in future experiments we have to select diagrams with more representative NDep metric values.

4. Comparison of results

An overall analysis of the obtained results (see tables 4 and 6) leads us to conclude that the metrics NC, NA, NM, NAssoc, NAgg, NGen, NAggH, NGenH, MaxHAgg, MaxDIT are to some extent correlated with the three maintainability sub-characteristics we considered. NDep seems to be the only non-correlated metric, although this preliminary result may be caused by the design of the experiment as in the second experiment the majority of the diagrams did not have dependencies. We believe it is too early to consider these results as definitive. As previously stated, further empirical validation is needed, including internal and external replication of these experiments, and also new experiments must be carried out, with practitioners who work in software development organisations. As Basili et al. [13] remark, after performing a family of experiments you can build the cumulative knowledge to extract useful measurement conclusions to be applied in real measurement projects.

Moreover, data related to "real projects" is also needed for gathering real evidence that these metrics can be used as early class diagram maintainability indicators.

5. Conclusions and future work

It is widely recognized that the quality of OO software must be assessed from the early phases of its development life cycle. This fact lead us to define a set of metrics for assessing the structural complexity of UML class diagrams, with the idea that they are correlated with the maintainability of such diagrams.

These early metrics could allow OO software designers a quantitative comparison of design alternatives, and therefore, an objective selection among several class diagram alternatives with equivalent semantic content, and the prediction of external quality characteristics, like maintainability in the initial stages of the IS life cycle and a better resource allocation based on these predictions. In this sense we have built prediction models (based on the metrics values) using two advanced techniques borrowed from artificial intelligence [30],[31]. Although the accuracy of prediction of those models is pending on further investigation.

Performing empirical validation with the metrics is fundamental in order to demonstrate their practical utility. In this line we have summarized two controlled experiments with the aim of corroborating if there is a significant correlation between the proposed metrics and the maintainability sub-characteristics: analysability, understandability and modifiability. The results obtained in both experiment shows that most of the metrics we proposed (NAssoc, NAgg, NaggH, MaxHAgg, NGen, NgenH and MaxDIT) are good indicators of class diagram maintainability sub-characteristics. We cannot, however, draw such firm conclusions regarding the NDep metric.

Some changes that could be made to improve the experiment presented are:

- Increase the size of the class diagrams. By increasing the size of the class diagrams we can have examples that are closer to reality. Also, as the examples are more realistic, and if we are working with professionals, we can make better use of their potential capability and conclude that the results are more general.
- Increase the difference between the values of the metrics. This option could lead to more conclusive results about the metrics and their relationship with the factor we are trying to control.
- Carry out the experiment in a more controlled environment.
- Work with real data. Another way to enhance the validity of the results is by working with real data obtained from industrial environments. However, the scarcity of such data continues to be a great problem so we must find other ways to tackle validating metrics.

Acknowledgements

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

References

[1] L. Briand , S. Arisholm, F. Counsell, F. Houdek and P. Thévenod-Fosse, "Empirical Studies of Object-Oriented Artefacts, Methods, and Processes: State of the Art and Future Directions", *Empirical Software Engineering,* 4(4), 1999, pp. 387-404.

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

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

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

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

[6] 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.

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

[8] Lorenz, M., and J. Kidd, *Object-Oriented Software Metrics: A Practical Guide,* Prentice Hall, Englewood Cliffs, New Jersey, 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] M. Genero, "Defining and Validating Metrics for Conceptual Models", *Ph.D. thesis,* University of Castilla-La Mancha, 2002.

[12] C. Calero, M. Piattini, and M. Genero, "Empirical validation of referential integrity metrics", *Information and Software Technology,* 43, 2001, pp. 949-957.

[13] V. Basili, F. Shull, and F. Lanubile, "Building Knowledge through Families of Experiments", *IEEE Transactions on Software Engineering,* 25(4), 1999, pp. 435-437.

[14] B. Kitchenham, S. Pfleeger, and N. Fenton, "Towards a Framework for Software Measurement Validation", *IEEE Transactions of Software Engineering,* 21(12), 1995, 929-943.

[15] N. Schneidewind, "Methodology For Validating Software Metrics", *IEEE Transactions of Software Engineering,* 18(5), (1992), 410-422.

[16] G. Cantone, and P. Donzelli, "Production and maintenance of software measurement models", *Journal of Software Engineering and Knowledge Engineering,* 5, 2000, pp. 605-626.

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

[18] L. Briand, C. Bunse, and J. Daly, "A Controlled Experiment for evaluating Quality Guidelines on the Maintainability of Object-Oriented Designs", *IEEE Transactions on Software Engineering,* 27(6), 2001, pp. 513-530.

[19] R. Harrison, S. Counsell, and R. Nithi, "Experimental Assessment of the Effect of Inheritance on the Maintainability of Object-Oriented Systems", *The Journal of Systems and Software,* 52, 2000, pp. 173-179.

[20] Wohlin, C., P. Runeson, M. Höst, M. Ohlson, B. Regnell, and A. Wesslén, *Experimentation in Software Engineering: An Introduction,* Kluwer Academic Publishers, 2000.

[21] D. Perry, A. Porter, and L. Votta, "Empirical Studies of Software Engineering: A Roadmap", *Future of Software*

Engineering, ACM, Ed. Anthony Finkelstein, 2000, pp. 345-355.

[22] V. Basili, and H. Rombach, "The TAME project: towards improvement-oriented software environments", *IEEE Transactions on Software Engineering,* 14(6), 1988, pp. 728-738.

[23] L. Godo, R. López de Mántaras, C. Sierra, and A. Verdaguer, "MILORD: the architecture and management of linguistically expressed uncertainty", *International Journal of Intelligent Systems,* 4, 1989, pp. 471-501.

[24] P. Bonissone, "A Fuzzy Sets Based Linguistic Approach: Theory and Applications", *Approximate Reasoning in Decision Analysis,* M. M. Gupta and E. Sanchez (eds.), North-Holland Publishing Company, 1982, pp. 329-339.

[25] http://department.obg.cuhk.edu.hk/ResearchSupport/Minimum_correlation.asp

[26] L. Briand, K. El Emam, and S. Morasca, "Theoretical and empirical validation of software product measures", *Technical Report ISERN-95-03,* International Software Engineering Research Network, 1995.

[27] G. Poels, and G. Dedene, "Distance-based software measurement: necessary and sufficient properties for software measures", *Information and Software Technology,* 42(1), 2000, pp. 35-46.

[28] Roberts, F., *Measurement theory with applications to decision making, utility, and the social sciences,* Addison-Wesley, Reading, MA, USA, 1979.

[29] A. Brooks, J. Daly, J. Miller, M. Roper, and M. Wood, "Replication of experimental results in software engineering", *Technical report ISERN-96-10,* International Software Engineering Research Network, 1996.

[30] M. Genero, J. Olivas, M. Piattini, and F. Romero, "Using metrics to predict OO information systems maintainability", *CAISE 2001, Lecture Notes in Computer Science, 2068,* Interlaken, Switzerland, 2001, pp. 388-401.

[31] M. Genero, L. Jiménez, and M. Piattini, "Empirical Validation of Class Diagram Complexity Metrics", *SCCC 2001,* IEEE Computer Society Press, Chile, 2001.