Does the Level of Detail of UML Models Affect the Maintainability of Source Code?

Ana M. Fernández-Sáez[1], Marcela Genero[2], and Michel R.V. Chaudron[3]

[1]Alarcos Quality Center, S.L., Department of Technologies and Information Systems, University of Castilla-La Mancha
Paseo de la Universidad 4, 13071, Ciudad Real, Spain
+34 926295300 ext.6648
ana.fernandez@alarcosqualitycenter.com

[2]ALARCOS Research Group, Department of Technologies and Information Systems, University of Castilla-La Mancha
Paseo de la Universidad 4, 13071, Ciudad Real, Spain
+34 926295300 Ext. 3740
Marcela.Genero@uclm.es

[3]LIACS - Leiden University
Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
+31 715277065 (secr 7061)
chaudron@liacs.nl

Abstract. This paper presents an experiment carried out as a pilot study to obtain a first insight into the influence of the quality of UML models on the maintenance of the corresponding source code. The quality of the UML models is assessed by studying the amount of information they contain as measured through a level of detail metric. The experiment was carried out with 11 Computer Science students from the University of Leiden. The results obtained indicate a slight tendency towards obtaining better results when using low level of detail UML models, which contradicts our expectations based on previous research found in literature. Nevertheless, we are conscious that the results should be considered as preliminary results given the low number of subjects that participated in the experiment. Further replications of this experiment are planned with students and professionals in order to obtain more conclusive results.

Keywords: UML, maintenance, empirical studies, controlled experiment

1 Introduction

The current increasing complexity of software projects [1] has led to the emergence of UML [2] as a tool with which to increase the understanding between customer and developer and to improve communication among team members [3]. Despite this, not all UML diagrams have the same complexity, layout, level of abstraction, etc. Previous studies have shown that the style and rigor used in the diagrams may vary

considerably throughout software projects [4], in addition to affecting the source code of the system in a different way.

On the one hand, the different purposes for which a model may be intended (for example: architecting solutions, communicating design decisions, detailed specification for implementation, or automatically generating implementation code) signifies that the same system can be represented with different styles. On the other hand, the development diagrams are sometimes available for maintainers, but this is not always the case, and the diagrams must be generated with a reverse engineering process. The difference in the origin of the models and the different techniques that can be used to generate a reverse engineering model result in different styles of models. Some of the most notable differences between these models may be the level of detail shown. In this work we therefore analyze whether the different levels of detail (LoD) affect the work that must be carried out by a maintainer.

This document is organized as follows. Section 2 presents the related work. Section 3 presents the description of the experiment. The results obtained in the experiment are presented in Section 4, whilst the threats to validity are summarized in Section 5. Finally, Section 6 outlines our main conclusions and future work.

2 Related work

We performed an SLR [5] to discover all the empirical studies performed as regards the use of UML in maintenance, and found only the following two works related to the maintenance of source code:

- In [6] an experiment was performed to investigate whether the use of UML influences maintenance in comparison to the use of only source code. The results of this work show a positive influence of the presence of UML for maintainers.
- In the work presented in [7], the experiment performed is focused on the comprehension and the difficulties involved in maintaining object-oriented systems. UML models were also presented to the subjects of the experiment, but they were only focused on exploring the participant's strategies and problems while they were conducting maintenance tasks on an object-oriented application.

We therefore decided to perform an experiment related to the influence of different levels of detail on UML diagrams when assisting in maintenance tasks. We found a paper [8] focused on the understandability of models with different LoD in the development phase. The results show a better understanding of models when they have a high LoD. We would like to discover whether high LoD diagrams help workers to perform the changes that need to be made to the source code during the maintenance phase.

3 Experiment description

The experiment was carried out at the University of Leiden (The Netherlands) in March 2011. In order to run and report this experiment, we followed the

recommendations provided in several works [9-11]. The experiment was presented by following the guidelines for reporting empirical research in software engineering [11] as closely as possible. The experimental material is available for downloading at: http://alarcos.esi.uclm.es/experimentUMLmaintenance/

In the following subsections we shall describe the main characteristics of the experiment, including goal, context, variables, subjects, design, hypotheses, material, tasks, experiment procedure and analysis procedure.

3.1 Goal

The principal goal of this experiment was to investigate whether the LoD in UML models influences the maintenance of source code. The GQM template for goal definition [12, 13] was used to define the goal of our experiment as follows: "*Analyze the level of detail in UML models with the purpose of evaluating it with respect to the maintainability of source code from the point of view of researchers, in the context of Computer Science students at the University of Leiden.*

As in [3], we considered that the LoD in UML models should be defined as the amount of information that is used to represent a modeling element. LoD is a 'continuous' metric, but for the experiment we have taken two "extremes" - high and low LoD.

We decided to use 3 different types of diagrams (use case, sequence and class diagrams) since they are those most frequently used. When the LoD used in a UML model is low, it typically employs only a few syntactical features, such as class-name and associations, without specifying any further facts about the class. When it is high, the model also includes class attributes and operations, association names, association directionality, and multiplicity. In sequence diagrams, in which there is a low LoD, the messages among objects have an informal label, and when the LoD is high the label is a method name plus the parameter list. We consider that it is not possible to distinguish between low and high LoD in use case diagrams because they are very simple diagrams. The elements that fit each level of detail are shown in Table 1.

Table 1. Levels of detail in UML models

Diagram	Element	Low LoD	High LoD
Class diagram	Classes (box and name)	✓	✓
	Attributes	✗	✓
	Types in attributes	✗	✓
	Operations	✗	✓
	Parameters in operations	✗	✓
	Associations	✓	✓
	Association directionalities	✗	✓
	Association multiplicities	✗	✓
	Aggregations	✓	✓
	Compositions	✓	✓
Sequence diagram	Actors	✓	✓
	Objects	✓	✓

	Messages in informal language	✓	✗
	Messages with formal language (name of a method)	✗	✓
	Parameters in messages	✗	✓
	Labels in return messages	✗	✓

3.2 Context Selection

The experimental objects consisted of use case, class and sequence diagrams and the JAVA code of two software systems, which are summarized below:

- A-H: high LoD diagrams and JAVA code of system A.
- A-L: low LoD diagrams and JAVA code of system A.
- B-H: high LoD diagrams and JAVA code of system B.
- B-L: low LoD diagrams and JAVA code of system B.

Diagrams A-x described a library domain from which a user can borrow books. Diagrams B-x described a sport centre domain from which users can rent services (tennis courts, etc.). System A is a Library extracted from [14]. We decided to use it because it was a representative system, it was complete (source code and models were available) and it gave us a starting point from which to compare our results (it was only possible to compare the results obtained from the subjects who received System A with high LoD with [7]). System B is a Sport centre application created as part of the Master's degree Thesis of a student from the University of Castilla-La Mancha, and we therefore consider it to be a real system. Both systems are desktop applications and have more or less the same complexity. These experimental objects were presented in English.

The subjects students on a Software Engineering course from which they had acquired training in UML diagrams. Their knowledge was sufficient for them to understand the given systems, and they had roughly the same background. They had knowledge about the use of UML diagrams in general, but they were taught about UML diagrams and JAVA in a training session organized to take place the day before the experiment was carried out.

The experiment was carried out by 11 Computer Science students from the University of Leiden (The Netherlands) who were taking the Software Engineering course in the second-year of their B.Sc.

Working with students also implies various advantages, such as the fact that their prior knowledge is fairly homogeneous, there is the possible availability of a large number of subjects [15], and there is the chance to test experimental design and initial hypotheses [16]. An additional advantage of using novices as subjects in experiments on comprehensibility and modifiability is that the cognitive complexity of the objects under study is not hidden by the subjects' experience. Nonetheless, we also wish to test the findings with practitioners in order to strengthen the external validity of the results obtained.

The students who participated in the experiment were volunteers selected for convenience (the students available in the corresponding course). Social threats

caused by evaluation apprehension were avoided by not grading the students on their performance.

3.3 Variables selection

The independent variable (also called "main factor") is the LoD, which is a nominal variable with two values (low LoD and high LoD). We combined each level of the independent variable with the two different systems used to obtain four treatments (see Table 2).

The dependent variables are modifiability and understandability. These two variables were considered because understandability and modifiability directly influence maintainability [17]. In order to measure these dependant variables, we defined the following measures:

- **Understandability Effectiveness (U_{Effec}):** This measure reflects the ability to correctly understand the system presented. It is calculated with the following formula: number of correct answers / number of questions. A higher value of this measure reflects a better understandability.
- **Modifiability Effectiveness (U_{Effic}):** This measure reflects the ability to correctly modify the system presented. It is calculated with the following formula: number of correctly performed modification tasks / number of modification tasks. A higher value of this measure reflects a better modifiability.
- **Understandability Efficiency (M_{Effec}):** This measure also reflects the ability to correctly understand the system presented. It is calculated with the following formula: time spent / number of correctly answered questions. A lower value of this measure reflects a better understandability.
- **Modifiability Efficiency (M_{Effic}):** This measure also reflects the ability to correctly modify the system presented. It is calculated with the following formula: time spent / number of correctly performed tasks. A lower value of this measure reflects a better modifiability.

Additional independent variables (called "co-factors") were considered according to the experimental design of the replication, and their effect has been controlled and analyzed:

- **Order**. The selected design (see Table 2), i.e., the variation in the order of application of each method (low LoD, high LoD), was intended to alleviate learning effects. Nonetheless, we analyzed whether the order in which the LoD were used by the subjects biased the results.
- **System**. This factor indicates the systems (i.e., A and B) used as experimental objects. The design selected for the experiment (see Table 2) forced us to choose two application domains in order to avoid learning effects. Our intention was that the system factor would not be a confounding factor that might also influence the subjects' performances. We therefore selected well-known domains and experimental objects of a similar complexity.

3.4 Hypotheses formulation

Based on the assumption that the more information a model contains, the more is known about the concepts/knowledge described in the model, the hypothesis are:

1. $H_{1,0}$: There is no significant difference in the subjects' understandability effectiveness when working with UML diagrams modeled using high or low levels of detail.
   $H_{1,1}$:$\neg H_{1,0}$
2. $H_{2,0}$: There is no significant difference in the subjects' understandability efficiency when working with UML diagrams modeled using high or low levels of detail.
   $H_{2,1}$:$\neg H_{2,0}$
3. $H_{3,0}$: There is no significant difference in the subjects' modifiability effectiveness when working with UML diagrams modeled using high or low levels of detail.
   $H_{3,1}$:$\neg H_{3,0}$
4. $H_{4,0}$: There is no significant difference in the subjects' modifiability efficiency when working with UML diagrams modeled using high or low levels of detail.
   $H_{4,1}$: $\neg H_{4,0}$

The goal of the statistical analysis will be to reject these null hypotheses and possibly to accept the alternative ones (e.g., H_{n1}=$\neg$ H_{n0}).

3.5 Experimental design

We selected a balanced factorial design in which the group-interaction acted as a confounding factor [18] which permits the lessening of the effects of learning and fatigue. The experiment's execution consisted of two runs. In each round, each of the groups was given a different treatment. The corresponding system (source code + UML models) was assigned to each group at random, but was given out in a different order in each case. Table 2 presents the outline of the experimental design.

Table 2. Experimental design

RUN 1		LoD		RUN 2		LoD	
		Low	High			Low	High
System	A	Group 1	Group 2	System	A	Group 3	Group 4
	B	Group 3	Group 4		B	Group 2	Group 1

Before carrying out the experiment, we provided the subjects with a background questionnaire and assigned them to the 4 groups randomly, based on the marks obtained in the aforementioned questionnaire (blocked design by experience) in an attempt to alleviate experience effects. To avoid a possible learning effect, the diagrams came from different application domains (A-a Library and B-a Sport centre).

When designing the experiment we attempted to alleviate several issues that might threaten the validity of the research done by considering the suggestions provided in [19].

3.6 Experimental tasks

The tasks to be performed did not require high levels of industrial experience, so we believed that the use of students could be considered appropriate, as suggested in literature [20, 21]. The material used was written in English.

There were three kinds of tasks:

- **Understandability task**: This contained 3 questions concerning the semantics of the system, i.e. the semantics of diagrams and the semantics of code. These questions were multiple choice questions and were used to obtain U_{Effec} and U_{Effic}.
- **Modifiability task**: The subjects received a list of requirements in order to modify the code of the system in order to add/change certain functionalities. This part of the experiment contained 3 modifiability tasks and allowed us to calculate M_{Effec} and M_{Effic}. The subjects were provided with answer sheets to allow them to structure their responses related to maintenance tasks. They had to fill in a different form depending on the element that they wished to maintain. The answer sheets can be found at:

  http://alarcos.esi.uclm.es/experimentUMLmaintenance/
- **Post-questionnaire task**: At the end of the execution of each run, the subjects were asked to fill in a post-experiment questionnaire, whose goal was to obtain feedback about the subjects' perception of the experiment execution, which could be used to explain the results obtained. The answers to the questions were based on a five-point Likert scale [22].

3.7 Experimental procedure

The experiment took place in two sessions of two hours each. The subjects first attended a training session in which detailed instructions on the experiment were presented and the main concepts of UML and JAVA were revised. In this session, the subjects carried out an exercise similar to those in the experimental tasks in collaboration with the instructor. During the training session, the subjects were required to fill in a background questionnaire. Based on the marks obtained in this questionnaire, the subjects were randomly assigned to the 4 groups shown in Table 2, thus obtaining balanced groups in accordance with the marks obtained in the background questionnaire.

The experiment then took place in a second session, consisting of two runs. In each run, each of the groups was given a different treatment, as is shown in Table 2.

The experiment was conducted in a classroom, where the students were supervised by the instructor and no communication among them was allowed.

After the experiment execution, the data collected from the experiment were placed on an excel sheet.

3.8 Analysis procedure

The data analysis was carried out by considering the following steps:

1. We first carried out a descriptive study of the measures of the dependent variables, i.e., understandability and modifiability.
2. We then tested the formulated hypotheses using the non-parametric Kruskal-Wallis test [23] for the data collected in the experiment. The use of this test was possible because, according the design of the controlled experiment, we obtained paired samples. In addition, Kruskal-Wallis is the most appropriate test with which to explore the results of a factorial design with confounded interaction [18, 24], i.e., the design used in our experiment, when there is non-normal distribution of the data.
3. We next used the Kruskal-Wallis test to analyze the influence of the co-factors (i.e., System and Order).
4. The data collected from the post-experiment questionnaire was finally analyzed using bar graphs.

4 Results

The following subsections show the results of the data analysis of the experiment performed using SPSS [25].

4.1 Descriptive statistics and exploratory analysis

Table 3 and Table **4** show the descriptive statistics of the Understandability and Modifiability measures, respectively (i.e., mean ($\overline{X}$), standard error (SE), and standard deviation (SD)), grouped by LoD.

Table 3. Descriptive statistics for U_{Effec} *and* U_{Effic}.

LoD	Subjects	U_{effec}			U_{Effic}		
		$\overline{X}$	SE	SD	$\overline{X}$	SE	SD
Low	N = 10 (1 outlier)	**0.767**	0.051	0.161	**334.500**	36.308	114.816
High	N = 11	0.758	0.650	0.215	363.924	82.602	273.960

Table 4. Descriptive statistics for M_{effec} *and* M_{Effic}.

LoD	Subjects	M_{effec}			M_{Effic}		
		$\overline{X}$	SE	SD	$\overline{X}$	SE	SD
Low	N = 11	**0.437**	0.066	**0.221**	**240.121**	41.008	136.007
High	N = 11	0.402	0.050	0.169	294.637	47.198	156.539

At a glance, we can observe that when the subjects used low LoD diagrams they obtained better values in all variables. This indicates that low LoD diagrams may, to some extent, improve the comprehension and modification of the source code.

4.2 Influence of LoD

In order to test the formulated hypotheses we analyzed the effect of the main factor (i.e. LoD) on the dependent variables considered (i.e., UEffec, UEffic, MEffec and MEffic) using the Kruskal-Wallis test (see Table 5).

Table 5. Kruskal-Wallis test results for U_{effec}, U_{effic}, M_{effec} and M_{effic}

	U_{effec}	U_{Effic}	M_{effec}	M_{Effic}
LoD	1	0.439	0.792	0.491

Testing $H_{1,0}$ (U_{Effec})
The results in Table 5 suggest that the null hypothesis cannot be rejected since the p-value is greater than 0.05. This means that there is no significant difference in U_{Effec} in either group.

We decided to investigate this result in greater depth by calculating the number of subjects who achieved better values when using the low LoD models (i.e. a low LoD value is higher than a high LoD value):

Table 6. Comparison of subjects' results for each measure

	low LoD = high LoD	low LoD < high LoD	low LoD > high LoD
U_{Effec}	6	3	2
U_{Effic}	0	7	4
M_{Effec}	0	7	4
M_{Effic}	0	5	6

As Table 6 shows, the number of subjects who obtained the same results for both treatments (high and low LoD) is relatively high. There were more subjects who performed better with a high LoD than with a low LoD, but the differences in comparison to the opposite group is very small (only one subject).

Testing $H_{2,0}$ (U_{Effic})
The results in Table 5 suggest that the null hypothesis cannot be rejected since the p-value is greater than 0.05. This means that there is no significant difference in U_{Effic} in either group.

We decided to investigate this result in greater depth by calculating the number of subjects who achieved better values when using the low LoD models (i.e. a low LoD value is smaller than a high LoD value):

As Table 6 shows, no subjects obtained the same U_{Effic} for both treatments (high and low LoD). More subjects performed better with a low LoD than with a high LoD.

Testing $H_{3,0}$ (M_{Effec})
The results in Table 5 suggest that the null hypothesis cannot be rejected since the p-value is greater than 0.05. This means that there is no significant difference in M_{Effec} in either group.

We decided to investigate this result in greater depth by calculating the number of subjects who achieved better values when using the low LoD models (i.e. a low LoD value is higher than a high LoD value):

As Table 6 shows, no subjects obtained the same M_{Effec} for both treatments (high and low LoD). More subjects performed better with a high LoD than with a low LoD.

Testing $H_{4,0}$ (M_{Effic})

The results in Table 5 suggest that the null hypothesis cannot be rejected since the p-value is greater than 0.05. This means that there is no significant difference in M_{Effic} in either group.

We decided to investigate this result in greater depth by calculating the number of subjects who achieved better values when using the low LoD models (i.e. a low LoD value is smaller than a high LoD value):

As Table 6 shows, no subjects obtained the same M_{Effic} for both treatments (high and low LoD). More subjects performed better with a high LoD than with a low LoD, but the differences in comparison to the opposite group are also small.

4.3 Influence of system

In order to test the effect of the co-factor System, we performed a Kruskal-Wallis test whose results are shown in Table 7. As all the p-values were higher than 0.05, except in one case (U_{Effic}), we did not have sufficient evidence to reject the hypothesis, i.e. it seems that the system did not influence the subjects' performance (and this was therefore a controlled co-factor).

Table 7. Kruskal-Wallis test results for the influence of the System.

	U_{effec}	U_{Effic}	M_{effec}	M_{Effic}
System	0.804	**0.035**	0.575	0.061

4.4 Influence of order

In order to test the effect of Order, we performed a Kruskal-Wallis test (see Table 8). As all p-values were higher than 0.05, we did not have sufficient evidence to reject the hypothesis, i.e. the order did not influence the subjects' performance (and this was therefore a controlled co-factor).

Table 8. Kruskal-Wallis tests results.

	U_{effec}	U_{Effic}	M_{effec}	M_{Effic}
Order	1	0.105	0.223	0.341

4.5 Post- experiment survey questionnaire results

The analysis of the answers to the post-experiment survey questionnaire revealed that the time needed to carry out the comprehension and modification tasks was

considered to be inappropriate (more time was needed), and that the subjects considered the tasks to be quite difficult (Fig. 1).

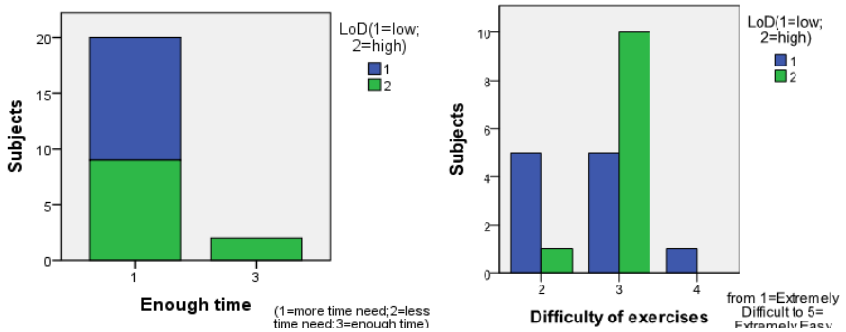

Fig. 1. Subjects' perception of the experiment.

We also asked about the subjects' perception of some of the items that appeared in the high LoD diagrams but did not appear in the low LoD diagrams. Fig. 2 shows that high LoD elements seem to be appreciated by the subjects. With regard to the histograms in Fig. 2, if a subject responds 1 or 2, this indicates that s/he thinks that the element in the question was helpful, while a response of 4 or 5 indicates that the elements in the question are not helpful (3 is a neutral response). If we focus on the elements related to class diagrams (upper histograms) we can see that attributes are helpful for 9 subjects (versus 1 subject who does not believe them to be helpful). The same is true of operations (10 subjects vs. 1 subject). If we focus on the elements related to sequence diagrams (lower histograms) we can see that formal messages are more helpful (16 subjects) than natural language messages (0 subjects), and the same can also be said of the appearance of parameters in messages (13 subjects vs. 2 subjects).

4.6 Summary and discussion of the data analysis

The null hypothesis cannot be rejected for any of the dependent variables. Although we cannot draw conclusive results on the main factor (LoD), we have found that co-factors (system, order) have not influenced the results.

Nevertheless, the descriptive statistics in general showed a slight tendency in favor of using low LoD diagrams in contrary to what we believed, as the diagrams with a high LoD helped developers in the software development stage [8]. This may result from the fact that the subjects did not have the expected amount of knowledge about UML (a mean of 8.8 correct answers out of 16 questions) and JAVA (a mean of 4.9 correct answers out of 9 questions) tested in the background questionnaire. The results of the experiment must be considered as preliminary results owing to the small size of the group of subjects who participated in the experiment.

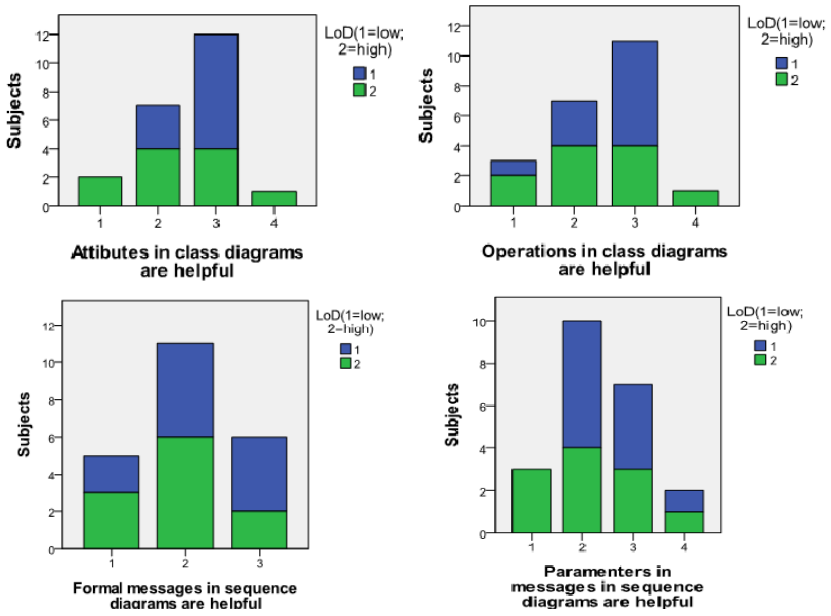

Fig. 2. Subjects' opinion of LoD (1=Complete Agreement 2=Partial Agreement 3=Neither agree/ nor disagree 4=Partial Disagreement 5=Total disagreement)

5 Threats to Validity

We must consider certain issues which may have threatened the validity of the experiment:

- **External validity**: External validity may be threatened when experiments are performed with students, and the representativeness of the subjects in comparison to software professionals may be doubtful. In spite of this, the tasks to be performed did not require high levels of industrial experience, so we believed that this experiment could be considered appropriate, as suggested in literature [13]. There are no threats related to the material used since the systems used were real ones.
- **Internal validity**: Internal validity threats are mitigated by the design of the experiment. Each group of subjects worked on the same system in different orders. Nevertheless, there is still the risk that the subjects might have learned how to improve their performances from one performance to the other. Moreover, the instrumentation was tested in a pilot study in order to check its validity. In addition, mortality threats were mitigated by offering the subjects extra points in their final marks.
- **Conclusion validity**: Conclusion validity concerns the data collection, the reliability of the measurement, and the validity of the statistical tests. Statistical tests were used to reject the null hypotheses. We have explicitly mentioned and discussed when non-significant differences were present. What is more,

conclusion validity might also be affected by the number of observations. Further replications on larger datasets are thus required to confirm or contradict the results.

- **Construct validity**: This may be influenced by the measures used to obtain a quantitative evaluation of the subjects' performance, the comprehension questionnaires, the maintenance tasks, and the post-experiment questionnaire. The metrics used were selected to achieve a balance between the correctness and completeness of the answers. The questionnaires were defined to obtain sufficiently complex questions without them being too obvious. The post-experiment questionnaire was designed using standard forms and scales. Social threats (e.g., evaluation apprehension) have been avoided, since the students were not graded on the results obtained.

6 Conclusions and future work

The main concern of the research presented in this paper is the use of a controlled experiment to investigate whether the use of low or high level of detail in UML diagrams influences the maintainer's performance when understanding and modifying source code. The experiment was carried out by 11 academic students from the University of Leiden in the Netherlands.

The results obtained are not significant owing to various factors such as the fact that the subjects selected had a low level of experience in using UML and JAVA code, and the small size of the group of subjects who participated in the experiment. It is only possible to observe a slight tendency towards obtaining better results with low LoD diagrams, contrary to the results obtained in [8].

Despite these drawbacks, we have ensured that the experimental results were not influenced by other co-factors such as the system used or the order in which the subjects received the experimental material.

We are planning to perform two replications with students from the University of Castilla-La Mancha (Spain) and students from the University of Bari (Italy). A third possible replication with professionals is also being planned. All the drawbacks found in the execution of this experiment will be taken into account in the replications.

Acknowledges. This research has been funded by the following projects: MEDUSAS (CDTI-MICINN and FEDER IDI- 20090557), ORIGIN (CDTI-MICINN and FEDER IDI-2010043(1-5), PEGASO/MAGO (MICINN and FEDER, TIN2009-13718-C02-01), EECCOO (MICINN TRA2009-0074), MECCA (JCMM PII2I09-0075-8394) and IMPACTUM (PEII 11-0330-4414).

References

1. Van Vliet, H., *Software Engineering: Principles and Practices* 3rd ed. 2008: Wiley.
2. OMG. *The Unified Modeling Language. Documents associated with UML Version 2.3* 2010; Available from: http://www.omg.org/spec/UML/2.3.
3. Nugroho, A. and M.R.V. Chaudron. *Evaluating the impact of UML modeling on software quality: An industrial case study*. in *Proceeding of 12th International*

Conference on Model Driven Engineering Languages and Systems (MODELS'09). 2009.
4. Lange, C.F.J. and M.R.V. Chaudron, *In practice: UML software architecture and design description.* IEEE Software, 2006. **23**(2): p. 40-46.
5. Fernández-Sáez, A.M., M. Genero, and M.R.V. Chaudron, *Empirical studies on the influence of UML in software maintenance tasks: A systematic literature review.* Submitted to Science of Computer Programming - Special issue on Software Evolution, Adaptability and Maintenance, Elsevier.
6. Dzidek, W.J., E. Arisholm, and L.C. Briand, *A realistic empirical evaluation of the costs and benefits of UML in software maintenance.* IEEE Transactions on Software Engineering, 2008. **34**(3): p. 407-432.
7. Karahasanovic, A. and R. Thomas. *Difficulties Experienced by Students in Maintaining Object-Oriented Systems: an Empirical Study*. in *Proceedings of the Australasian Computing Education Conference (ACE'2007)* 2007.
8. Nugroho, A., *Level of detail in UML models and its impact on model comprehension: A controlled experiment.* Information and Software Technology, 2009. **51**(12): p. 1670-1685.
9. Juristo, N. and A. Moreno, *Basics of Software Engineering Experimentation*. 2001: Kluwer Academic Publishers.
10. Wohlin, C., et al., *Experimentation in Software Engineering: an Introduction.* 2000: Kluwer Academic Publisher.
11. Jedlitschka, A., M. Ciolkowoski, and D. Pfahl, *Reporting Experiments in Software Engineering*, in *Guide to Advanced Empirical Software Engineering* F. Shull, J. Singer, and D.I.K. Sjøberg, Editors. 2008, Springer Verlag.
12. Basili, V. and D. Weiss, *A Methodology for Collecting Valid Software Engineering Data.* IEEE Transactions on Software Engineering, 1984. **10**(6): p. 728-738.
13. Basili, V., F. Shull, and F. Lanubile, *Building Knowledge through Families of Experiments.* IEEE Transactions on Software Engineering, 1999. **25**: p. 456-473.
14. Eriksson, H.E., et al., *UML 2 Toolkit*. 2004: Wiley.
15. Verelst, J. *The Influence of Abstraction on the Evolvability of Conceptual Models of Information Systems*. in *International Symposium on Empirical Software Engineering (ISESE'04)*. 2004.
16. Sjøberg, D.I.K., et al., *A Survey of Controlled Experiments in Software Engineering.* IEEE Transaction on Software Engineering, 2005. **31**(9): p. 733-753.
17. ISO/IEC, *ISO/IEC 25000: Software Engineering*, in *Software product quality requirements and evaluation (SQuaRe)*. 2008, International Organization for Standarization.
18. Kirk, R.E., *Experimental Design. Procedures for the Behavioural Sciences*. 1995: Brooks/Cole Publishing Company.
19. Wohlin, C., et al., *Experimentation in Software Engineering: An Introduction*. 2000, Norwell, MA, USA: Kluwer Academic Publishers.
20. Basili, V., F. Shull, and F. Lanubile, *Building Knowledge through Families of Experiments.* IEEE Transactions on Software Engineering, 1999. **25**(4): p. 456-473.
21. Höst, M., B. Regnell, and C. Wholin. *Using students as subjects - a comparative study of students and professionals in lead-time impact assessment*. in *4th*

Conference on Empirical Assessment and Evaluation in Software Engineering. 2000.
22. Oppenheim, A.N., *Questionnaire Design, Interviewing and Attitude Measurement*. 1992: Pinter Publishers.
23. Conover, W.J., *Practical Nonparametric Statistics*. 3rd ed. 1998: Wiley.
24. Winer, B.J., D.R. Brown, and K.M. Michels, *Statistical Principles in Experimental Design*. 3rd ed. 1991: Mc Graw Hill Series in Psychology.
25. SPSS, *SPSS 12.0, Syntax Reference Guide*. 2003, Chicago, USA: SPSS Inc.