A research plan for gathering empirical evidence of the benefits of using UML in software maintenance

Ana M. Fernández-Sáez[1], Marcela Genero[2], 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
ana.fernandez@alarcosqualitycenter.com
[2]ALARCOS Research Group, Department of Technologies and Information Systems, Paseo de la Universidad 4, Ciudad Real 13071, Spain
Marcela.Genero@uclm.es
[3]LIACS - Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands
chaudron@liacs.nl

Abstract. This paper presents a research proposal for gathering empirical evidence of the benefits of using the Unified Modelling Language (UML) in software maintenance. The main research questions, which drive our research through empirical research methods, are presented and are briefly explained along this work.

Keywords: UML, research methods, empirical studies, software maintenance.

1 Introduction

Due to the increasing complexity of software projects nowadays [16], UML [15] emerges as a tool for increasing the understanding between customer and developer (in the analysis phase), for improving the communication among team members [10] and for increasing the understanding of how software works (both in the development and the maintenance phase).

Despite this, from an economic point of view, any type of investment must be justified in terms of how much payback there will be at a later stage. That being the case, in the context of software projects, investing in modelling should be justified by benefits, such as improved productivity and improved product quality that can be gained later during software development or maintenance. This is one of the main reasons for investigating whether the use of UML can generate important differences that would make the costs involved worthwhile. This is particularly true in the context of software maintenance, which consumes the majority of software development resources, as explained in [12] and [4]: “Maintenance typically consumes 40 percent to 80 percent of software costs. Therefore, it is probably the most important life cycle phase of software” and “60 percent of the budget is spent on software maintenance, and 60 percent of this maintenance is to enhance". Enhancing old software is, therefore, a big business.

The main goal of this paper is to present a long-term research plan which we propose to develop to investigate whether the use of UML provides any benefit in software maintenance tasks or not (see Section 2). This is an in progress work; for that reason part of the plan is now being executed and the other part is pending to be dealt with in the future.

2 Research plan

The main goal of our research is to investigate the benefits of modelling on software maintenance tasks. In particular, we focus our attention on UML [15] as a modelling language because UML is a widely used modelling language in industry. To achieve the goal, three initial research questions have been defined (see Figure 1). The research questions outlined below are formulated to look at the impacts of UML modelling on software maintenance from different perspectives (e.g., from the point of view of software engineers and from empirical data obtained from industrial software projects). Apart from the different perspectives, new insights obtained during the study could also lead to the formulation of new research questions. Several research strategies are employed to address those questions. By considering different angles and conducting multiple research strategies in answering the grand question, we expect to obtain a more comprehensive understanding about the impacts of UML modelling on software maintenance.

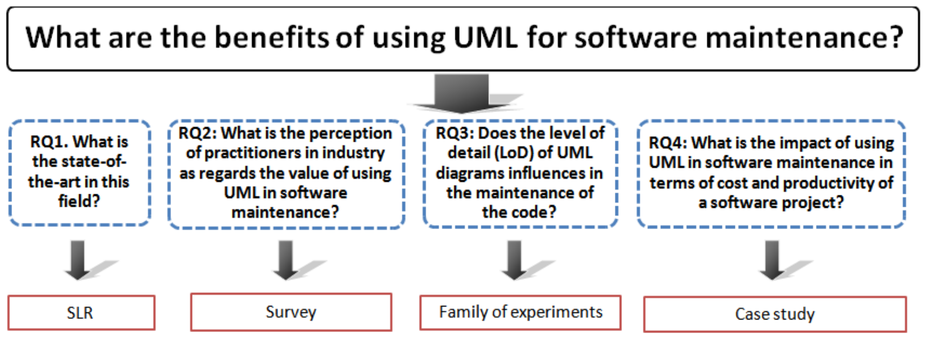
What are the benefits of using UML for software maintenance?
RQ1. What is the state-of-the-art in this field?
RQ2: What is the perception of practitioners in industry as regards the value of using UML in software maintenance?
RQ3: Does the level of detail (LoD) of UML diagrams influences in the maintenance of the code?
RQ4: What is the impact of using UML in software maintenance in terms of cost and productivity of a software project?
SLR
Survey
Family of experiments
Case study

Fig. 1. Research plan.

As our approach for addressing the research questions (see Figure 1) of this study is empirical in nature, in this section we will explain the empirical research methods we propose to use to get empirical evidence that allows to answers the proposed research questions.

For answering RQ1 we decided to carry out a systematic literature review (SLR) as is suggested in [1]. This SLR allowed us to analyze what has already been done in this field and to identify the remaining gaps which are potential areas for further investigation. The formulation of the other RQs was based on the results of the SLR.

RQ2, addressed by means of interviews [14] and a survey [11], will provide us with a subjective idea of the success of using UML in software maintenance and how and why maintainers feel the benefits or drawbacks of the use of UML in their daily tasks. RQ4, addressed by case studies [13] will provide us with an industrial point of view concerning the influence of UML in maintenance tasks. Whilst waiting for getting industrial case studies that would allow us to work on RQ4, we decided to tackle RQ3 by performing a family of experiments [17]. We first considered performing an experiment to discover whether or not the use UML models benefits maintenance tasks, but the results of RQ1 showed us that this topic had already been studied in [2]. We began performing some interviews related to the RQ2, and we found that maintenance tasks are sometimes supported by UML diagrams built from the beginning of the development and in other cases the diagrams are obtained through reverse engineering processes.. The different origins of the diagrams make the diagrams have different LoD. This fact led us to decide to continue working on RQ3 through a family of experiments, for investigating whether or not the LoD of the UML diagrams influences on maintenance tasks. To our knowledge the influence of LoD has already been studied but related to the development stage [8, 9], not to maintenance.

2.1 RQ1: What is the state-of-the-art in this field?

For answering RQ1, we carried out a SLR [3]. The main goal of this SLR is to gather the existing empirical evidence about the influence of UML models in software maintenance as seen in six digital sources (Scopus, Science Direct, Wiley InterScience, IEEExplore, ACM, Springer), from 1997 to March 2010. Through this SLR we saw what has already been done in this field, also identifying the remaining gaps which are potential areas for further investigation. These gaps are intended to be covered by the research questions presented in the previous section. This SLR discovered 53 relevant papers in peer-reviewed journals, conferences, and workshops (they can be found in http://alarcos.esi.uclm.es/SLR-UMLinMaintenance/) and classified these in order to obtain responses to the research questions presented and summarized briefly below (note that the percentages are referring to the total number of found studies):

SLR Research Question 1 asked, *Which diagrams are used most in studies into UML and maintenance?* The results show a clear ordering which indicates the relative importance that researchers attach to 3 diagram types: class diagrams (32.00% of primary studies), statechart diagrams (27.20%) and sequence diagrams (14.40%). The low occurrence of papers relating to the use case diagrams (6.40%) could be explained by the fact that there are no studies addressing this type of diagrams which are in turn directly related to maintenance tasks. This small amount of papers could also be related to the origin of the models. In some cases the models are obtained from the code, using reverse engineering. In this case the use case diagrams are generally not available. Furthermore, use cases say nothing about the structure of the system; hence they do not contain information that a maintainer needs for performing changes/modifications.

SLR Research Question 2 asked, *Which variables are investigated in the empirical studies (experiments, case studies and surveys)?* Most of the studies that were found are experiments (96.70%), focusing their efforts on measuring the understandability of the UML models. The variables which are measured in these papers are thus related to the time (25.7%) and the correctness (11.17%) obtained in the test by the subjects. There are some more isolated studies which focused on the influence of UML models on maintenance tasks where the variables measured are, apart from time, the correctness of the proposed solutions and the quality of the code.

SLR Research Question 3 asked, *What is the state-of-the-art in empirical studies of UML maintenance?* To answer this research question, an analysis based on different perspectives of the empirical literature in the field is presented. The analysis is presented from the following four perspectives: How?, Where?, Who? and What?

- *How is the influence of UML in maintenance studied?* Most of the studies found present result of controlled experiments (96.70%). This is a well-known way to validate data. However, the field would benefit (in terms of generalizability) from also performing case studies.
- *Where are the empirical studies carried out?* These studies are carried out in a laboratory context (76.92%), so it is also necessary to perform more empirical studies in industrial contexts to corroborate the academic results.
- *Who is evaluated in the empirical studies?* The subjects who performed the tests are mostly students (80.58%). A minority of papers involves academic staff (11.65%) or practitioners (7.77%).
- *What is maintained in the empirical studies?* We obtained that most of the studies are related only to the maintenance of UML models (88.37%), instead of the UML models and the code (11.63%). Also it is important to highlight that most of the models used represent prototypes of systems or very simple systems (71.83%). So, performing more studies with real industrial systems is necessary.

The main results of this SLR indicate that the external validity of the empirical studies, in majority of experiments, is questionable, given the material, tasks and subjects used. That being so, there is a need for more empirical studies such as experiments and case studies executed in industrial contexts, i.e. with real systems, maintenance tasks performed by practitioners under real conditions, to gather real empirical evidence of the influence of UML in maintenance.

2.2 RQ1: What is the perception of industry professionals as regards the value of using UML in the maintenance?

To address RQ1, we are planning to perform a web-based survey. We are not looking for a specific role as subject, since we wish to discover the perception of UML from different perspectives, but the results will be grouped by roles or groups of roles. The questions in the survey aim to discover what type of documentation is actually being used for maintenance and, whenever UML models are used (it may occur that models are available but not used), we wish to know which models are used most frequently. Furthermore, we deem it important to ascertain whether in the industrial environments in which UML is used as a tool to perform the maintenance tasks, there is a person who is responsible for updating the models or whether, on the other hand, the models are not updated.

A small amount of semi-structured interviews have already been conducted with practitioners in Dutch software companies who are involved in maintenance projects in which UML is used. Interviews have also been carried out with staff in India (via videoconference), because some of the companies contacted outsource maintenance work to this country. Given that managing to interview subjects from different countries is extremely difficult because software maintenance is carried out in geographically distributed projects and an interview is a time-consuming task, we used the results from interviews to create the survey that we wish to perform.

Given that managing to get subjects from different countries to be interviewed is extremely difficult, as is gathering evidence from those situations, we are now contacting people from Spain, and Italy.

2.3 RQ2: Does the level of detail (LoD) of UML diagrams influences in the maintenance of the code?

To address RQ2, we are planning to carry out a family of experiments (see Figure 2), whose main goal is to "*analyze the level of detail in UML models for the purpose of evaluating it with respect to the maintainability of systems from the point of view of the researcher, in the context of students in Computer Science and professionals*".

There are two independent variables: the LoD with two values (low LoD and high LoD) and the diagram domain (A and B). By combining each level of the independent variables we obtain four treatments (see Table 1). The objects of study are software systems (source code + UML models). The UML diagrams considered in this experiment are use case diagrams, sequence diagrams and class diagrams.

As in [10] we considered that the LoD in UML models is defined as the amount of information that is used to represent a modeling element. When the LoD used in a UML model is low, it typically employs only a few syntactic 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, where there is 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. In use case diagrams with low LoD only actors, use cases and relations among them are represented in the model, but in a high LoD model there are also extended and included use cases.

The dependent variable is maintainability measured through effectiveness (number of correctly performed modification tasks /number of modification tasks) and efficiency (number of correctly performed tasks/time).

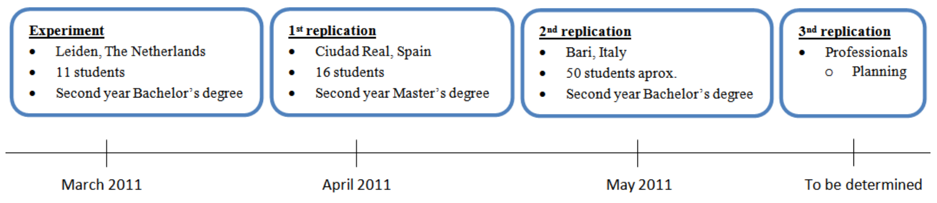

Fig. 2. Chronology of the family of experiments.

Based on the assumption that the higher the amount of information put into a model, the more is known about the concepts/knowledge described in the model the hypotheses are: *1) H_{01}: There is no significant difference of subjects' effectiveness when working with UML diagrams modeled using high or low level of detail. H_{11}:¬H_{01}, 2) H_{02}: There is no significant difference of subjects'efficiency when working with UML diagrams modeled using high or low level of detail. H_{11}: ¬H_{02}*

Before experiment execution we provide the subjects with a background questionnaire and assign the subjects to the 4 groups randomly based on the marks obtained in the aforementioned questionnaire (blocked design by experience). In this way we try to alleviate experience effects. The experiment execution will consist of two runs. In each round, each of the groups will be given a different treatment. We will assign the corresponding system (source code + UML models) to each group at random, but will give them out in a different order in each case. Table 1 presents the outline of the experimental operation. This assignment of treatments corresponds with the selected balanced within factorial design with group-interaction confounding, which permits the lessening of the effects of learning and fatigue.

Table 1. Experiment runs

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

2.4 RQ3: What is the impact of using UML in software maintenance in terms of cost and productivity of a software project?

To address RQ3, we are planning to conduct case studies in industrial environments. We cannot, as yet, provide details of the design of the case studies, because it is still in its early stages. We wish to investigate whether the investment in modelling in maintenance software projects is justified by benefits, such as improved productivity and improved product quality.

We are still in the process of contacting software companies to obtain case studies that fit our purposes (projects using UML in maintenance tasks).

Acknowledgements. 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. Arisholm, E., L.C. Briand, S. E. Hove, Labiche, Y.: The impact of UML documentation on software maintenance: An experimental evaluation. IEEE Transactions on Software Engineering. Vol. 32(6), 365-381 (2006)
2. Dzidek,W.J., E. Arisholm, Briand L.C.: A realistic empirical evaluation of the costs and benefits of UML in software maintenance. IEEE Transactions on Software Engineering. Vol. 34(3), 407-432 (2008)
3. Fernández-Sáez, A.M., Genero, M., Chaudron, M.R.V.: Empirical studies on the influence of UML in software maintenance tasks: A systematic literature review. Submitted to Elsevier SCICO - Special issue on Software Evolution, Adaptability and Maintenance.
4. Glass R.: Facts and Fallacies of Software Engineering. Addison-Wesley (2002)
5. ISO/IEC, ISO/IEC 25000: Software engineering - Software product quality requirements and evaluation (SQuaRe), International Organization for Standarization (2008)
6. Kitchenham, B., Charters, S.: Guidelines for performing systematic literature reviews in software engineering. Keele University: EBSE-2007-01 (2007)
7. Nugroho, A., Chaudron M.R.V.: A survey into the rigor of UML use and its perceived impact on quality and productivity. In Proceedings of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, ESEM'08, Kaiserslautern, Germany (2008)
8. Nugroho, A., Flaton, B. and Chaudron, M. R. V.: Empirical Analysis of the Relation between Level of Detail in UML Models and Defect Density. Springer-Verlag, City (2008)
9. Nugroho, A.: Level of detail in UML models and its impact on model comprehension: A controlled experiment. Information and Software Technology, Vol. 51(12), 1670-1685 (2009)
10. Nugroho, A., Chaudron M.R.V.: 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).
11. Pfleeger, S., Kitchenham, B.: Principles of survey research: part 1: turning lemons into lemonade. ACM SIGSOFT Software Engineering Notes. Vol. 26(6), 16-18 (2001)
12. Pressman, R.S.: Software Engineering: A Practitioners Approach, seventh ed. McGraw Hill, (2005)
13. Runeson, P., Höst, M.: Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering. Vol. 14(2), 131-164 (2009)
14. Steinar, K.: Doing interviews. SAGE Publications (2007)
15. Object Management Group: The Unified Modeling Language. Documents associated with UML Version 2.3 http://www.omg.org/spec/UML/2.3 (2010)
16. Van Vliet, H.: Software Engineering: Principles and Practices (3rd Edition). Wiley (2008)
17. Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., Wesslén, A.: Experimentation in software engineering: an introduction. Kluwer Academic Publishers, Norwell, MA, USA (2000)