BENEVOL 2010

December 16
Lille, France

9th edition of the
WORKSHOP
BElgian-NEtherlands software eVOLution seminar

Proceedings

http://rmod.lille.inria.fr/benevol

INRIA lifL Université Lille1 Sciences et Technologies cnrs dépasser les frontières GPL Génie de la Programmation et du Logiciel GdR Groupement de Recherche

Organization

Organizing Committee

Nicolas Anquetil, University Lille 1 & INRIA, France
Mark van den Brand, University of Eindhoven, The Netherlands
Anthony Cleve, University Lille 1 & INRIA, France
Anne-Françoise Le Meur, University Lille 1 & INRIA, France
Kim Mens, University Louvain la Neuve, Belgium
Sébastien Mosser, University Lille 1 & INRIA, France
Damien Pollet, University Lille 1 & INRIA, France
Romain Rouvoy, University Lille 1 & INRIA, France

Program Committee

Stéphane Ducasse, University Lille 1 & INRIA, France (co-chair)
Laurence Duchien, University Lille 1 & INRIA, France (co-chair)
Lionel Seinturier, University Lille 1 & INRIA, France (co-chair)
Antoine Beugnard, Telecom Bretagn,e France
Xavier Blanc, University of Bordeaux, France
Mark van den Brand, University of Eindhoven, The Netherlands
Serge Demeyer, University of Antwerp, Belgium
Dirk Deridder, Vrije Universiteit Brussel, Belgium
Eric Dubois, Centre Henri Tudor, Luxemburg
Patrick Heymans, University of Namur, Belgium
Robert Hirschfeld, Hasso-Plattner Institut Potsdam, Germany
Andy Kellens, Vrije Universiteit Brussel, Belgium
Günter Kniesel, University of Bonn, Germany
Paul Klint, CWI, The Netherlands
Pierre Kelsen, University of Luxemburg, Luxemburg
Tom Mens, University of Mons, Belgium
Martin Monperrus, Darmstadt University of Technology, Germany
Mario Südholt, École des Mines de Nantes & INRIA, France
Andy Zaidman, Technische Universiteit Delft, The Netherlands

Table of Contents

EVOLUTION IN CBSE AND SERVICES APPROACHES

Investigating the benefits of UML on software maintenance: A research proposal

Ana M. Fernández Sáez[a,b], Marcela Genero[a], Michel R. V. Chaudron[b],

[a]*ALARCOS Research Group, Department of Technologies and Information Systems, Paseo de la Universidad 4, Ciudad Real 13071, Spain*
[b]*LIACS - Leiden University, Niels Bohrweg 1, 2333 CA Leiden, The Netherlands*

Abstract

This paper presents a research proposal on how and why we will investigate the influence of the Unified Modeling Language (UML) on software maintenance tasks. The main goal is to present the main research questions, along with an explanation of which research methods we propose to use to obtain the answers to those questions empirically.

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

1. Introduction

Due to the increasing complexity of software projects nowadays [14], UML [13] emerges as a tool for increasing the understanding between customer and developer (in the analysis phase), for improving the communication among team members [8] 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 modeling 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 [10] 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 [13] 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. 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.

- *RQ1*. What is the perception of practitioners in the industry as regards the value of using UML in software maintenance?

- *RQ2*. Does the level of detail (LoD) of UML diagrams influence software maintenance?

- *RQ3*. What is the impact of using UML in software maintenance in terms of cost and productivity of a software project?

As our approach for addressing the research questions of this study is empirical in nature, in this section we will detail the empirical research methods we propose to use to get empirical evidence that allows to answers the proposed research questions. As is suggested in [6] we begin collecting the existing empirical evidence through a systemaic litearture review. Later we plan to carry out several empirical studies considering the main emprical strategies suggested in [15]: survey [9], interview [12] (as a way to perform surveys), case study [11], and experiment [15].

2.1. Preliminary study: A Systematic Literature Review (SLR)

Firstly, we began our research carrying out a SLR [3] 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 refering 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%), statechart diagrams (27.20%) and sequence diagrams (14.40%). Some studies performed partial comparisons of the understandability of one type of diagram versus another. The three aforementioned diagram types are reported to contribute most to understandability. 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 wich focused on the influence of UML models on maintenance tasks. In these cases, 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 RQ3, 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?
  This is related to the previous question, from which we know that most of the studies perform controlled experiments. 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?
  If we focus on the results obtained in this SLR, we can see 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.

It is also noteworthy that there are only two papers directly related to empirical studies on the use of UML in maintenance tasks. The first of them is [2] which presents the results of two controlled experiments carried out with students from different universities. [1] presents the results of a controlled experiment carried out with practitioners. In both cases, the time taken to perform the modifications to the system, the time spent on maintaining the models and the quality of the proposed modifications are measured. [2] reports the result that the time taken to make changes in the system is lower when one is using UML models than when they are not used, while if it includes time to perform the corresponding modifications on models there is no significant difference. However, in both cases the quality of the modification is greater when the subjects have UML models. In contrast, study [1] does not find any significant difference in the time spent on performing changes, but they obtain the result that the quality of the changes is higher for the group of subjects with UML models, as is the case in [2].
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.
We have conducted a further study related to extracting the different factors that can influence the maintainability of systems from the studies analyzed in this SLR. The factors are shown in single-bordered boxes 1, and in double-bordered boxes there are some categories that we added to classify all the factors. A factor that has a positive influence is represented with the symbol +, and the negative influence with −. Because of the fact that understandability directly influences maintainability [5], we are assuming here that those factors that are related to the understandability are also related to the maintainability. The content of Figure 1 is explained thus:

- The maintainability of a system is influenced by the maintainability of its code and its models.

- The maintainability of the code is negatively influenced when the complexity of the system is high, but it is positively influenced by the experience or ability of the maintainers and also by a correct traceability from the models to the code.

- The maintainability of the UML models is positively influenced by the availability of class, sequence, and statechart diagrams, and also if the models contain stereotypes that detail certain characteristics of its elements.

- The use of composite states improves the understandability of UML statechart diagrams.

- A high nesting level of composites state in UML statechart diagrams negatively influence the understandability of the models.

- The availability of interactive views or animations to improve the diagrams improves the visualization of a UML model, thereby improving its maintainability.

- A proper distribution of the elements improves the maintainability of a UML model.

- The level of detail in UML models also affects the maintainability of systems (code and models), making it ideal to have a higher level of detail.

How the classification process of the factors was carried out is detailed below. Maintainability of a whole can be influenciated by the maintainabilities of each of its parts (code, models, documentation, etc.). In this case, we considered only code and models as part of a system.

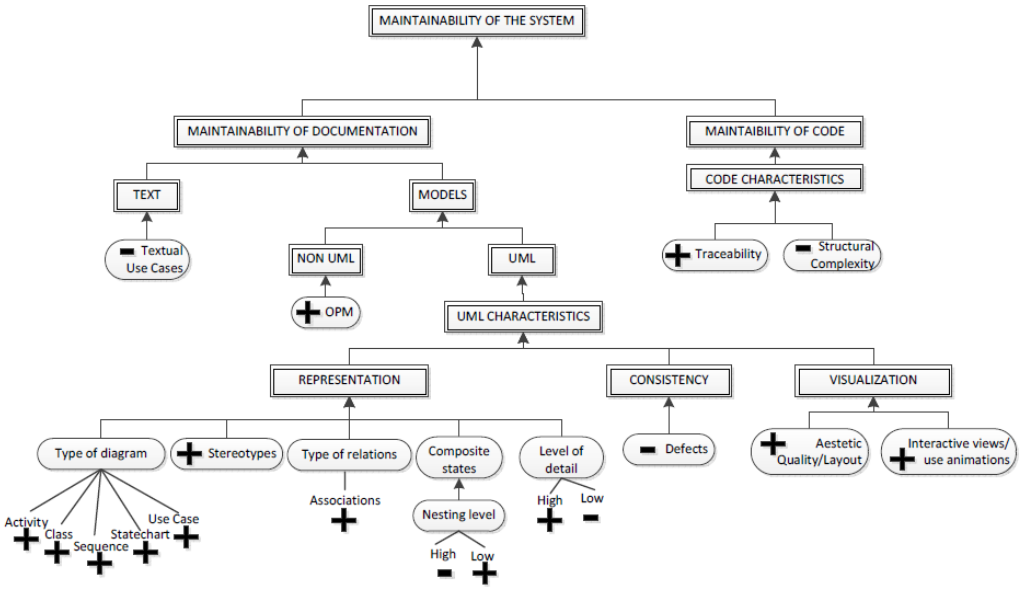

Figure 1: Factors that influence the maintainability of a system.

The maintainability of the models consists of some characteristics directly related with the model and ohers that the reader of the model introduces. A model can be influenced by it representation (the diagram by itself, what is represented) and the way that is presented to the reader, i.e. its visualization.
The maintainability of the code is influenced by its own characteristics, as well as by the characteristics from the models (because they are a complementary information about the code). The reader of the code also brings to it some influencing factors. To know which paper refer to each factor visit the webpage http://alarcos.esi.uclm.es/SLR-UMLinMaintenance/

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 carrying out a series of semi-structured interviews with practitioners in software companies who are involved in maintenance projects in which UML is used.We are not looking for a specific role as interviewee, because we would like to know about the perception UML from different perspectives.
The questions in the interview aim at finding out what type of documentation is actually being used for maintenance and, whenever UML models are used (it may happen that models are available but not used), we want to know which models are used most. Furthermore, it seems important to know whether in 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 wheteher on the other hand nobody updates them. The interview process is described below. An audio-recording is made of all interviews. The next step involves transcribing and coding the audio-recordings. All separate statements made by the subject are collected in a list. We then mark each statement that was related to the use of UML. After this initial coding process, we group the statements and identify (formulate) a common idea that best described all statements in one group. We then confronted the interview participants with this distilled list to validate our interpretations. We repeated the coding process and update the coding in the same way.
Up to now, we have conducted a few interviews in Dutch software companies. We have also done an interview with staff from India (via videoconference), because some of the companies contacted outsource maintenance work to this country.
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 controlled experiment, 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 second year undergraduate students in Computer Science from the University of Leiden, The Netherlands".

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 [8] 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).

This work differs from that presented in [2] in that it investigates whether or not the availability of UML diagrams influences the correctness and effectiveness of code maintenance, so the independent variable is different (LoD vs. UML model availability). In the case of the work presented in [8], the independent variable is the same but the dependent variable is different because that research focused on the understandability of the system, while we focus on maintainability.

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:

H_0: There is no significant difference of system maintainability between subjects when working with UML diagrams modeled using high or low level of detail. H_1: $\neg H_0$

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 groupswill be given a different treatment. As mentioned previously, in order to alleviate learning effect, we will divide the experimental subjects into 4 groups. 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. We are planning to carry out this experiment next February.

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

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

Table 1: Experiment runs.

We are also considering potential replications with students at the University of Bari (Italy) and in the University of Castilla-La Mancha, Ciudad Real (Spain), and if possible, to strengthen the results, we will perform the experiment with practitioners. After gathering evidence from this family of experiments we plan to integrate the results through meta-analysis techniques.

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 provide details of the design of this empirical study, because it is in the early stages as yet. We are still contacting software companies to get case studies that fit our purposes.

3. Conclusions

This paper focuses on presenting a research proposal about how investigating the influence of the use of UML on software maintenance. Also, some early results are presented.

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 and MECCA (JCMM PII2I09-0075-8394).

References

[1] Arisholm, E., L.C. Briand, S. E. Hove, and Y. Labiche, The impact of UML documentation on software maintenance: An experimental evaluation. *IEEE Transactions on Software Engineering*, 2006. 32(6): p. 365-381.

[2] 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.

[3] Fernández-Sáez, A.M., Genero, M., and Chaudron, M.R.V. Empirical studies on the influence of UML in software maintenance tasks: A systematic literature review. LIACS Technical Report 2010-05, University of Leiden, 2010.

[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. and S. Charters, Guidelines for performing systematic literature reviews in software engineering. 2007, *Keele University: EBSE-2007-01*.

[7] Nugroho, A. and M.R.V. Chaudron. 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)*, 2008. Kaiserslautern, Germany.

[8] 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.

[9] Pfleeger, S., and Kitchenham, B. Principles of survey research: part 1: turning lemons into lemonade. *ACM SIGSOFT Software Engineering Notes*, 2001. 26(6): p. 16-18.

[10] Pressman, R.S., Software Engineering: A Practitioners Approach, seventh ed. *McGraw Hill*, 2005.

[11] Runeson, P., and Höst, M. Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering*, 2009. 14(2): p. 131-164.

[12] Steinar, K. Doing interviews. *SAGE Publications*, 2007.

[13] Object Management Group. The Unified Modeling Language. Documents associated with UML Version 2.3 http://www.omg.org/spec/UML/2.3

[14] Van Vliet, H. Software Engineering: Principles and Practices (3rd Edition). *Wiley*, 2008

[15] Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., and Wesslén, A. Experimentation in software engineering: an introduction. *Kluwer Academic Publishers*, Norwell, MA, USA, 2000.