Exploring Costs and Benefits of Using UML on Maintenance: Preliminary Findings of a Case Study in a Large IT Department

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

[1]ALARCOS Research Group, Instituto de Tecnologías y Sistemas de Información,
University of Castilla-La Mancha, Spain
ana.fernandez@alarcosqualitycenter.com,
Marcela.Genero@uclm.es

[2] Joint Computer Science and Engineering Department,
Chalmers University of Technology & University of Gothenburg, Sweden
chaudron@chalmers.se

Abstract. UML has become the de-facto standard for graphical modelling of software. One source of resistance to model-based development in software organizations is the perception that the use of UML is not cost-effective. It is important to study what costs and benefits are experienced in industrial use, and in what context. In this paper we pay special attention to the maintenance phase, because maintenance consumes a significant part of software project resources. This paper describes a case study in an industrial context: the software department of a large multinational company. This case study presents qualitative analysis based on 2 0 out of 36 interviews performed with employees who played different roles in the company and provided different views about the use of UML. The results revealed that the investment needed for using UML in a company is relatively small and that it is mostly related to tooling and training. The principal use of UML diagrams is communication. The use of UML diagrams is also found to be related to fewer software defects. The costs of UML use should not be considered as a high investment. The paybacks of using UML are a better understanding of the problem domain, improved communication, reduction of software defects, improvement in software quality or reduction of software maintenance effort.

Keywords: UML, Software Maintenance, Modelling Languages, Case Study

1 Introduction

Modelling is a common aspect of effective software engineering, and UML is the de-facto standard notation for this. How to do software modelling effectively is still an open question. Given that a large portion of software development effort is spent on software maintenance [1], it is important to understand the impact of software modelling on software maintenance. In this paper, the term "*maintenance*" refers to those projects that modify or correct existing systems instead of creating new ones, i.e., the

focus is on repairing bugs and on creating new releases. In this study we explicitly aim to elicit factors related to the costs of using modelling, thus adding fresh findings to the hitherto scarce evidence on payoffs and costs of software modelling.

The principal goal of our research is to find out what industrial software professionals perceive as costs and benefits of software modelling, with special attention to software maintenance tasks. We focus our attention particularly on UML as a specific modelling language, because it is widely used in industry[2, 3]. In this paper we present empirical evidence obtained in the IT department of a large multinational company. This evidence was collected over a 12-month period in 2012.

Using the Goal-Question-Metrics template, we can formulate the goal of this study as follows: "***Analyze*** *the use of UML modelling* ***for the purpose of*** *investigating its costs and benefits,* ***with respect to*** *software maintenance tasks,* ***from the perspective of*** *the researcher,* ***in the context of*** *a large IT department".*

We wish to investigate whether the investment in UML is justified by benefits in software maintenance projects, such as improved productivity and improved product quality. We define the following research questions:

RQ1) What is the cost of using UML in software maintenance projects?

RQ2) What is the payback of using UML in software maintenance projects?

This paper is organized as follows. Section 2 presents the related work. Section 3 describes the case study and how it was designed. The results obtained are set out in Section 4, whilst the summary is provided in Section 5. Finally, Section 6 outlines our main conclusions and future work.

2 Related Work

After carrying out a Systematic Literature Review (SLR) [4] and later extending the search period till August 2013, we found 6 experiments related to the use of UML on the maintenance of source code. Only 2 experiments, using professionals as subjects, were discovered [5, 6], which concluded that the correctness or quality of the maintenance of the code is improved when UML diagrams are available, although the time of maintenance is not influenced. Related to the results obtained in academic environments with students, the results of Scanniello et al. [7] revealed that the availability of UML diagrams produced in the design phase positively influence the performance of maintenance tasks. But on the other hand, the presence of UML analysis diagrams does not show a clear influence on the understandability and modifiability of the source code[8]. This means that the phase in which the diagrams are created is an influential factor. But, is that difference based on the Level of Detail (LoD) presented in the diagrams? It seems that a higher LoD UML diagram improves the understanding and modifiability of source code compared to lower LoD UML diagrams, but the differences are not conclusive [9]. Focusing on the origin of the UML diagrams, in [10] we found that there is a clear preference for human-created diagrams (built during the development phase) over those generated using automatic reverse engineering tools, because they reduce the reading problems. The difference in performance is not significant, however.

The pattern that emerges from the results of these experiments is that, under controlled conditions, both students and professionals benefit to some extent from the use of UML in software maintenance. An important issue is to study if these results also hold in an industrial environment under real conditions. Pursuing this goal, we carried out the case study described in this paper.

3 Case Study Design and Execution

In this section, we discuss underlying aspects of the case study, following the suggestions provided in the literature for that purpose[11].

3.1 Specific Research Questions

It is difficult to measure the payback and costs of the use of UML precisely, because there is much noise in project administrations. We chose to aim for qualitative findings by performing interviews with different roles (software engineers, testers, developers, etc.). We broke down the research question further into the following:

1. What are the costs related to UML tooling? This question is related to RQ1.
2. What are the costs related to UML training? This question is related to RQ1.
3. What is the impact of UML diagrams on software maintainers' understanding and product quality? This question is related to RQ2.

3.2 Case and Subject Selection

For our case study we obtained data in an IT department of a multinational company. The IT department has between 800-1000 employees. In this department most projects are mainly of a software maintenance character. Following the classification of Yin[12], our study is a single, embedded case study. Our units of analysis are the different roles.

3.3 Data Collection Procedures

To obtain data about the use of UML during maintenance tasks we used two sources:

- **Department shared project files:** The IT department has a file server in which all the relevant documentation of the department and the projects is shared. Through these shared files the maintenance projects shares the project documentation and relevant documentation of the IT department.
- **Company personnel**: The researcher himself, as a temporary member of the organization and in the capacity of research intern, had direct access to the company staff and, in particular, to the people involved with the maintenance projects.

Using the first source, we obtained the quantitative data related to the investment carried out by the company for the introduction or improvement of UML modelling.

We also obtained qualitative data by interviewing personnel. We used semi-structured interviews[1] where the interviews are "guided conversations"[13]. The interviews are standardized, in the sense that each interviewee is asked similar questions, yet they are also open-ended, in that there is ample room for interviewees to elaborate.

3.4 Case Study Execution and Analysis Procedure

We performed 36 interviews of about one hour each, which were recorded and transcribed. We analysed each transcription, highlighting the important and surprising statements, using the NVivo tool. After that, we coded the statements and grouped them under more general themes. The interviews were performed with people of different roles, to obtain different points of view. The interviewee roles include: project managers, information analysts, project architects, technical lead, programmers or application developers, test engineers, delivery leads, SCRUM masters, system analysts.

4 Results

In this section we present the highlights from the findings of the study, based on the analysis of 20 of the 36 interviews. However, we already saw saturation of findings; hence we do not expect many new findings from fresh analysis.

4.1 What Are the Costs Related to UML Tooling?

We made an inventory of the tools in use in the company: Visio (15% of people using a modelling tool), Bizz Design Architect (5%) and Sparxs Enterprise Architect (80%), taking into account that one person might use more than one tool. The prices of licenses of these tools are between 135€ and 160€; a total of 150 licenses were needed in an IT department of 800-1000 employees. In addition, an amount of between 4,000€ and 6,500€ per year was paid as maintenance costs related to the use of the tools.

Although the tools used are part of the "expensive range" of tools, their costs are very small, relatively, compared to the yearly budget (mostly in manpower) of software maintenance projects. Moreover, the costs of tooling are fixed and can be paid off fast.

4.2 What Are the Costs Related to UML Training?

To answer this question, we used historical data provided by the person who manages internal/external training and courses for employees at the company; this data was from 2006 to May 2012. We selected those courses which were related to training on UML and separated them from other related topics (like Object Orientation, RUP,

[1] The interview questions can be found at: http://alarcos.esi.uclm.es/download/list-of-questions.pdf.

etc.), but sometimes those topics are taught together. Those courses usually take one week (40 hours approximately), and they do not have a learning test at the end of them.

The total amount of money spent by the company in UML adds up to 24,313€ in a period of 6 and a half years (which is approximately 3,750€ per year). Again, as for tooling, this amount is small, compared to the total budget of the department.

4.3 What Is the Impact of UML Diagrams on Software Maintainers' Understanding and Product Quality?

To answer this question, we performed interviews with different people involved in software maintenance projects. We present the results grouped by topic in the following subsections. The percentages presented below indicate the percentage of interviewees that mention this term/topic.

UML Usage.

The UML diagrams which the interviewees mentioned that they usually use during maintenance are the following: sequence diagrams (80% of interviewees), class diagrams (60%), activity and use case diagrams (50%), deployment diagrams (40%), component diagrams (30%) and collaboration diagrams (10%). These diagrams are used during the whole maintenance process, from the requirements specification starting with the design of use case diagrams, to the deployment of the system maintained in the operation environment using the deployment diagrams.

Purpose of Use of UML.

One of the questions during the interview was: "*Why do you use UML diagrams? / For what purpose is UML modelling used?*" The answers to these questions were varied. The majority of people use UML as a communication tool (22%). This communication can be between team members, including stakeholders (8%), or members of other teams (5%). UML is also used to communicate the current situation to newcomers to the project (7%). The broad use of UML as a representation for communication might be due to its being a standard notation, and also because it is well-known, both by professionals and recent graduates. At the same time, people recognize that UML diagrams are used to complement verbal communication (face to face or written), but not to replace it: "*[…] UML helps to improve the communication, but it doesn´t replace it […]*".

The next most common uses of UML diagrams are for: enhancing people's own understanding of the system under maintenance (8%), analysing risks (7%) and guiding testing (7%). Less-often mentioned are possible uses for: getting an overview (5%) or guiding implementation (5%).

Uses that were mentioned, but only rarely (2-3%), include: documenting, following the mandatory process, justifying costs, planning, supporting maintenance, determining responsibilities for success (offshore team), monitoring implementation, professional way of developing, or showing progress.

Finally, we should remark that some possible purposes which we expected to find were not actually mentioned by any of the interviewees, like certification, deployment, generation of implementation, knowledge transfer or reasoning about design.

Cost of Using UML.

We also asked the interviewees about the possible cost factors or investment related to the use of a modelling notation like UML in a software maintenance company: "*What cost factors are related to using UML modelling in your work?*"

Table 1 shows the responses to this question. The majority of those interviewed consider training as an important investment. This might be due to a fear of their own poor understanding of UML. Another investment which is often mentioned by interviewees is the cost of migration of the current situation to the new one, especially in the documentation. Formally speaking, this is related more to the introduction of UML than to the use of UML, yet it is potentially a major investment. Most comments related to migration came from people who are currently working on non-UML projects, and who would like to introduce it, but they consider the migration of the documentation to be an impassable hurdle.

Table 1. Cost factors related to the use of UML.

Cost factor	% references
Training	33%
on UML notation	22%
on modelling tool	5%
Migration	28%
Change of people's mind	11%
Tooling	11%
Central governance	5%
Learning curve	5%
Change of process	5%

Advantages and Disadvantages of UML.

We also asked the interviewees about the perceived advantages and disadvantages of the use of UML diagrams: "*Do you think UML has advantages? What are these? And disadvantages?*" The results are shown in Table 2.

Note that "high level of abstraction" is mentioned as an advantage and a disadvantage at the same time. This may be because architects feel abstraction is beneficial, but developers need diagrams which are closer to the source code.

We should take into account that the majority of the advantages commented, especially those related to the UML characteristics, are not benefits in themselves. They can, however, be considered as benefits in comparison with other modelling languages.

Some of the disadvantages mentioned (like "No semantics", "Unclear syntactics", "Difficulties in understanding the notation") might be caused by a poor understanding of UML diagrams. This problem could be solved by providing training in UML to users who do not feel comfortable with employing it.

Table 2. Advantages and disadvantages of UML.

Advantages	Disadvantages
Related to UML characteristics	
High level of abstraction High suitability for designing OO systems Shows different points of view Standardized	Not executable No/Unclear Semantics Freedom in styles - naming - layering... High level of abstraction Lack of user's point of view Low capability of designing SOA No enforcement for separation of what and how
Related to UML usage	
Helps to clarify procedures Helps in structuring the way of modelling Improves documentation Is a common language - world acceptance Is the only modelling language learnt properly Reduces misunderstandings/ gaps in offshoring	Difficulties in understanding the notation Difficulties modelling complex things Not enough expressiveness

UML Usage and the Quality of Software.

We asked the interviewees about the quality of the final product and its relationship with the use of UML diagrams: "*Do you think UML helps to improve the quality of the final product? How?*"

In this case interviewees considered quality of source code related to performing correct testing and obtaining positive results from it; i.e., obtaining a source code aligned with requirements and design: *"[…] Quality is the result of checking the result also, so UML is your reference of what this should be, but you have to check if the code that is delivered is in fact aligned with your UML diagram. […]"*

Employees of projects which are not using UML diagrams commonly believe that the presence/absence of diagrams is related to high/low quality of documentation, respectively. It is very important to note that there is universal agreement amongst all interviewees that the use of UML improves the software quality (100%).

In relation to software quality, we also asked the interviewees about the possible relationship between the use of UML diagrams and the presence of defects in the code of the system: "*Do you think that the use of modelling introduces errors?*"

17% of the interviewees considered that UML usage reduces the introduction of defects in the code of the system, i.e., prevents defects, while 8% believed that UML increases them. 8% of those interviewed think that there is no relation between software defects and UML in itself; the defects are caused by an incorrect solution, but UML is not the problem. Almost half of the interviewees (42%) are of the opinion that the use of UML is helpful when we need to find the cause of a problem in the source code.

Standardization.

We asked the interviewees about standardization in ways of working. In this case, we focussed on those standards used to document the system and the activity of diagramming. Only 10% of the interviewees considered that there is excessive standardisation, while 37% believed that there is a lack of standardization. These last respondents felt a need for more standardization related to the following:

- **Naming**: naming conventions for classes, attributes, etc. in code and diagrams.
- **Layering**: it is not clear what the recommended layering of the system is.
- **Style**: There are a lot of issues related to the style of diagramming (and subsequently of coding) which are not clear.
- **Level of detail**: it is not clear at what level of detail systems should be modelled.

Independently of their opinion on the presence of standards at the company, most of those interviewed (53%) agreed that there is a lack of conformance to the standards. Mechanisms to incentivise the correct use of standards should thus be introduced: "*If you let people choose, you lose all your advantages. So, yes, force them.*"

5 Threats to Validity

We must consider certain issues which may threaten the validity of the case study[11]:

- **Internal validity**: The age, education, role or experience of the interviewees might be influential factors in being for, or against, the use of UML. This factor will be analysed in future work.
- **External validity**: the sample of the case study might be a threat to the validity of this study, although the sampling process was as randomized as possible. The generalization of the results might be extended to cases which have common characteristics.
- **Construct validity**: the transcript of interviews and observations were sent back to the interviewees to enable correction of raw data. Apart from that, analyses were presented to them and to the internal research supervisor, in order to maintain their trust in the research.
- **Reliability**: the chain of evidence from the interviews and documentation analyzed through to the synthesized evidence was maintained using a word-for-word transcription (so as not to reach mistaken interpretation while the analysis was being undertaken; this analysis took a long time to carry out). Tools were also used during the analysis of the data. In addition, randomized pieces of the analysis were discussed by the researchers, so that they could verify and reach an agreement on them.

6 Conclusions and Future Work

This work aimed to discover the costs and benefits of using UML modelling in the setting of maintenance-intensive software development.

In an effort to answer the first two research questions of this study, we have reported on the costs of use and introduction of UML modelling. In the context of a large IT department these costs related to tooling and training can be considered relatively small. In addition, the cost of building the UML documents is considered as low by the majority of interviewees. The cost of maintenance of the UML documents is zero, due to the fact that in the majority of cases the UML documents are not synchronized with the updates performed in the source code. The payback of UML use is very difficult to measure, because one of the main benefits is the improvement of communication between stakeholders. That is why we decided to investigate the impact of UML diagrams on software maintainers' understanding and product quality as a third research question. We therefore asked employees for their subjective opinion of the use of UML diagrams, as well as about their benefits. As on all issues, there are those in favour and those against the use of UML, but we detected more people in favour of using it. Proponents of modelling could be found within project architects, developers and maintenance engineers. Opponents to modelling could be found in Agile formation and people who are less familiar with UML. We speculate that people who are opposed to UML modelling are individuals who have been working at the company for a very long time, who are used to working in a certain way and thus are fearful of change.

Several benefits have been reported regarding the use of UML: better understanding of the problem domain, improved communication, reduction of SW defects, improvement in quality or reduction of software maintenance effort. We would recommend strengthening the benefits mentioned in the employees' ideas, also introducing the rest of the possible advantages to them (like reducing rework, improving the requirements, a better understanding of the solution space, etc.).

As part of the analysis of the costs and paybacks of the modelling during maintenance, several additional issues were detected, which should be dealt with in the company in the quest to improve the maintenance process. There is a need for standardization – which should focus in particular on the style of modelling: 1) Naming and layering conventions should be defined; and 2) The level of detail which should be presented on diagrams should be defined.

A very important issue which must be improved is the need to keep diagrams and the documentation in-synch with source code, representing on these all the changes performed in the system. In order to keep the diagrams updated, we recommend the use of a version management tool of diagrams. In relation to this topic, we observed that the process and responsibility for updating the documentation is often not clearly assigned. Finally, we recommend incentivizing or giving training on the long term benefits of using modelling languages (especially UML) to those subjects who do not know them and who cannot feel there is any possible benefit from a change in the process. People should also be incentivized regarding the benefits of maintaining the documentation.

Nevertheless, we will continue analysing the remaining interviews, in order to corroborate the results obtained. The analysis of the documentation of each project and its relation with employees' opinion will also be done as part of future work.

Acknowledgements

This research has been funded by the GEODAS-BC project (Ministerio de Economía y Competitividad and Fondo Europeo de Desarrollo Regional FEDER, TIN2012-37493-C03-01).

References

1. Pressman, R.S.: Software engineering: a practitioners approach. McGraw Hill (2005).
2. Dobing, B., Parsons, J.: How UML is used. Communications of the ACM. 49(5), 109–113 (2006).
3. Scanniello, G., Gravino, C., Tortora, G.: Investigating the Role of UML in the Software Modeling and Maintenance - A Preliminary Industrial Survey. Presented at the International Conference on Enterprise Information Systems (2010).
4. Fernández-Sáez, A.M., Genero, M., Chaudron, M.R.V.: Empirical studies concerning the maintenance of UML diagrams and their use in the maintenance of code: A systematic mapping study. Information and Software Technology. 55(7), 1119–1142 (2013).
5. Dzidek, W.J., Arisholm, E., Briand, L.C.: A realistic empirical evaluation of the costs and benefits of UML in software maintenance. IEEE Transactions on Software Engineering. 34(3), 407–432 (2008).
6. Arisholm, E., Briand, L.C., Hove, S.E., Labiche, Y.: The Impact of UML Documentation on Software Maintenance: An Experimental Evaluation. IEEE Transaction on Software Engineering. 32(6), 365–381 (2006).
7. Scanniello, G., Gravino, C., Tortora, G.: Does the Combined use of Class and Sequence Diagrams Improve the Source Code Comprehension? Results from a Controlled Experiment. Presented at the Experiences and Empirical Studies in Software Modelling Workshop (2012).
8. Scanniello, G., Gravino, C., Genero, M., Cruz-Lemus, J.A., Tortora, G.: On the Impact of UML Analysis Models on S ource Code Comprehensibility and Modifiability. ACM Transactions On Software Engineering And Methodology (In press) (2013).
9. Fernández-Sáez, A.M., Genero, M., Chaudron, M.R.V.: Does the Level of Detail of UML Models Affect the Maintainability of Source Code? Presented at the Experiences and Empirical Studies in Software Modelling Workshop (2012).
10. Fernández-Sáez, A.M., Chaudron, M.R.V., Genero, M., Ramos, I.: Are forward designed or reverse-engineered UML diagrams more helpful for code maintenance?: a co ntrolled experiment. Presented at the International Conference on Evaluation and Assessment in Software Engineering (2013).
11. Runeson, P., Höst, M., Rainer, A., Regnell, B.: Case Study Research in Software Engineering: Guidelines and Examples. Empirical Software Engineering, 14, 131-164 (2012).
12. Yin, R.K.: Case Study Research: Design and Methods. SAGE Publications (2002).
13. McNamara, C.: General guidelines for conducting interviews. Authenticity Consulting, LLC, Minneapolis, MN (1999).