On the Use of UML Documentation in Software Maintenance: Results from a Survey in Industry

Ana M. Fernández-Sáez
Institute of Technologies and Information Systems,
University of Castilla-La Mancha
Ciudad Real, Spain
anamaria.fernandez.saez@gmail.com

Marcela Genero
Institute of Technologies and Information Systems,
University of Castilla-La Mancha
Ciudad Real, Spain
marcela.genero @uclm.es

DaniloCaivano
Department of Informatics, University of Bari,
Bari, Italy
caivano@di.uniba.it

Michel R.V. Chaudron
Joint Computer Science and Engineering Department,
Chalmers University of Tech. & University of Gothenburg,
Gothenburg, Sweden
chaudron@chalmers.se

***Abstract*—This paper presents the findings of a survey on the use of UML in software maintenance, carried out with 178 professionals working on software maintenance projects in 12 different countries. As part of long-term research we are carrying out to investigate the benefits of using UML in software maintenance, the main objectives of this survey are: 1) to explore whether UML diagrams are being used in software industry maintenance projects; 2) to see what UML diagrams are the most effective for software maintenance; 3) to find out what the perceived benefits of using UML diagrams are; and 4) to contextualize the kind of companies that use UML documentation in software maintenance. Some complementary results based on the way the documentation is used (whether it is UML-based or not) during software maintenance are also presented.**

***IndexTerms*—UML, Software Maintenance, Survey.**

I. INTRODUCTION

UML [1] has become the de facto standard modeling notation used as a graphical notation to complement software documentation. It would therefore be useful for the software industry to study whether or not the use of UML benefits software maintenance, particularly because any type of investment must be justified from an economic point of view; i.e., there should be a payback at a later phase. In the context of software projects, investment in modelling should thus be justified by benefits (such as improved productivity and better product quality) that can be gained later, during software development or maintenance.

To the best of our knowledge, there are only three empirical studies on the impact of UML documentation on software maintenance in industry. Dzidek et al. [2] presented an experiment using 20 professional developers as subjects. Scanniello et al. [3] showed the results of an exploratory survey to investigate the state of the practice regarding the use of UML in software development and maintenance, with 22 employees at Italian companies. Fernández-Sáez et al. [4]presented an industrial case study performed in a large ICT (Information and communication technologies) department, in which 20 ICT professionals were interviewed. Their aim study was to investigate the use of UML diagrams during software maintenance. Summing up the results of these 3 studies, it would appear that this notation is frequently used, at least in Italy. This might be because using UML during maintenance seems to be beneficial in terms of software quality, even though no time is saved.

Although there is some encouraging evidence concerning the benefits of using UML during software maintenance in industry, it is scarce and based on a small population. It would therefore be useful to go one step further and collect a larger population of evidence from industry. Case studies in industrial contexts typically take a long time; it is difficult to obtain a large population of projects in industry that provide appropriate data. We have therefore attempted to bridge this gap by carrying out a survey of 178 ICT professionals from 12 countries (across the globe); that survey will be presented in this work as a first approach to the status of the current industrial environments. We decided to use this method because surveys are well-established social science techniques that can be used to gather information and opinions from a large population known to be representative of a target population [5]. Online-surveys may have some limitations, such as sampling bias and difficulties in designing clear, unbiased and unambiguous questionnaire items. But we have attempted to mitigate these as far as possible, by involving external researchers and some employees in the review and improvement of the experimental material, and by taking into account similar studies in the software engineering community.

The main goal of this survey is to address the following research questions (RQs):

- RQ1: Is UML documentation used to support software maintenance in industry?
- RQ2: What are the perceived benefits of using UML during software maintenance?
- RQ3: What kinds of companies use UML documentation during software maintenance?

- RQ4: What subset of UML diagram types has been demonstrated to be most effective during software maintenance?

The remainder of the paper is organized as follows: Section II describes the main steps of the survey design, while the results of the survey are presented in Section III. The threats to validity are set out in Section IV. Finally, our conclusions and future work are presented in Section V.

II. Survey Description

The survey was designed and reported by following the recommendations provided in [6].

A. Goal and Research Questions

The main goal of this survey is to address the RQs presented in Section I.

B. Target Population, Sample Identification and Recruitment Strategy

Our target population consisted of practitioners who have worked on maintenance projects, whether or not they have used UML. We considered ICT companies that develop, maintain or sell software as a principal part of their business, or companies focused on other business but with a large ICT department. The selection of the companies (sampling) was conducted by using the network contacts of the research groups of the authors of this paper who conducted the survey. Each author defined his or her own list of contacts, which included:

- employees at companies involved in research projects with the authors' universities, or that host students from the authors' universities for internships or thesis projects;
- the authors' former students, now employed at software companies;
- researchers from other universities with whom the authors have collaborated;
- people from professional networks or companies included in public-private research of which the authors' universities are members.

Upon receiving a filled-in survey, we asked the respondent to provide us with more contacts. A single list of 585 contacts was eventually obtained and used to distribute the survey. We also advertized the survey in software maintenance communities on the Internet, on sites such as the International Software Engineering Research Network, (which is for people who follow the International Conference on Software Maintenance), or the site of Software Maintenance and Reengineering.

C. Survey Structure

The survey was structured in blocks which grouped the questions into four topics:

1. *Demographic information*: this refers to information about the person replying such as: gender, educational qualifications, country in which they work, role in the company, experience in ICT and experience in software maintenance. This block of 7 questions helped us to contextualize the responses obtained.
2. *Organizational information*: the objective was to characterize the respondent's company. In particular, we collected information concerning: the size of the ICT department, stability of the maintenance team, or whether the company is geo-distributed or co-located. This block contained 6 questions designed to answer part of RQ3.
3. *Project information*: this refers to information regarding the most typical projects carried out in the company. The block included questions related to items such as: size of systems maintained, size of maintenance teams or type of maintenance carried out, and contained 4 questions related to RQ3. It is important to define the types of maintenance mentioned in this paper. They were divided into the following categories [7]: 1) *Corrective maintenance tasks*, i.e., those related to fixing a bug, 2) *Adaptive maintenance tasks*, i.e., those related to the changes made to the hardware/software platform, interface or requirement in order to improve performance or conform better to the law, or changes in the operative context; and 3) *Evolutive maintenance tasks*, i.e., those related to the development of new functionalities or functional/technical requirements requested by a customer.
4. *Process information*: this consisted of questions in which we asked whether the respondents create UML during software development, and if so, of what type; we also asked about their use during software maintenance. This block contained 11 questions related to RQ1, RQ2 and RQ4.

D. Survey Design

To address the research questions formulated, we drew up a survey consisting of 4 blocks of questions, with 28questions in all. Some questions were not presented to all individuals, as they were determined by the responses provided to other questions (i.e., conditional ones). Each person therefore answered a maximum of 22 questions. The electronic copy of the survey and the questionnaire flow is available online at: http://alarcos.esi.uclm.es/ShortSurvey-UML-Maintenance/

Most of the questions were measured using Likert scales, and a few others were measured with nominal scales, but they were all closed questions. Some of them also included a space for extra information, however. To avoid bias, the questions were ordered in such a way that the answer to one question would not affect the answers to the following ones. Though originally designed in English, Spanish and Italian versions were also used.

E. Survey Construction and Execution

The procedure followed consisted of the following steps:

1. An initial set of questions was selected by using similar surveys (such as those in [8], [9]) as a basis and tailoring them to our goals. A list of possible contacts was created by following the recruitment strategy explained above.
2. A pilot study with five industrial ICT professionals from an Italian company was performed before the survey was made available online. This was to refine it and to reduce any ambiguities, and minor changes were then made to the survey.

3. The survey was online from February to April of 2013, using Survey Monkey [10].
4. Contacts were invited (via email or phone) to participate the study. A reminder was sent to those who had committed to completing the survey, but who had not returned it by the end of March.
5. After the surveys had been collected, analyses were performed, aiming to answer the research questions. Data analysis was based on a quantitative analysis focusing mainly on descriptive statistics and percentages of the information collected.

III. RESULTS

A total of 268 ICT practitioners of the 585 directly contacted showed interest in responding to the survey. We filtered some of these, because, although they were interested in collaborating, they did not have the profile intended for this survey; i.e., they did not work in software maintenance. In the end, 178 responded to the survey during the two months it was online. This result is significant because of the difficulty normally involved in obtaining such a large quantity of individuals suitable for making up a target population. Importantly, no money or other incentives were given to the respondents, making the very high number of responses for a study of these characteristics in ICT environments even more surprising. We cannot state how many different companies these 178 people represent, as for reasons of privacy we did not ask the respondents to indicate their company. Please bear in mind that the responses to all the questions, summarized in the following subsections, do not add up to 178 in all cases, since some people did not answer all the questions(because of some conditional questions).

A. Overview and Descriptive Statistics

When analyzing the demographic information from the survey, we attempted to describe the respondents' profiles. 78% of them (139) were male and 22% were female (39), which was the proportion expected based on our personal perception of typical proportions for ICT. The countries in which the participants work are very varied: Afghanistan (1), Algeria (1), Austria (2), Canada (1), China (4), Finland (1), India (2), Italy (110), Mexico (1), Netherlands (18), Spain (23) and Uruguay (14).

The majority of those taking part have a high educational level with a Master's Degree (43%), or a medium high level with a Bachelor's Degree (33%); 4% have a researcher profile (they have PhD studies) and 17% have high school studies only. These percentages allow us to state that the sector is mature in terms of skilled professionals.

In terms of experience, the majority of the respondents (71%) are experienced professionals with more than 5 years in the area of ICT, and only 2% have less than 1 year of experience in this field. The other 26% of the survey participants possess between 1 and 5 years of experience. If we focus on experience in the field of software maintenance, then the percentages change: over half the respondents have more than 5 years of experience; only 6% have less than 1 year. The results lead us to assume that some experience is needed in the ICT field in general before working in the software maintenance field. This may be due to the need to have sufficient experience in understanding how systems are built before being able to modify them. We could have excluded those respondents who did not have very much experience in software maintenance; we decided against this, in order to obtain a real representation of industrial workers. The role most frequently played by the participants is that of programmer/coder (34%), followed by software analyst (19%), and project manager (15%). The remaining roles (business analyst, designer, software architects, software testers, etc.) are performed by less than 8% each. 66% of the respondents are currently working on software maintenance, while the others (34%) have worked on it in the past. In most cases (61%), those who replied perform maintenance on software which was developed by the same company, as against the 37% who maintain software developed by third parties.

The demographic results are consistent with those found in [11], i.e., most UML users are highly-educated and experienced. They also play a variety of roles, but most of them are software developers.

If we filter out those who do or do not use UML diagrams during software maintenance by role (*Fig.1*), we can state that the roles that use UML diagrams most frequently are software architects (as expected, because they create them), followed by software analysts and project managers. Project managers may be UML consumers [13]. Those who use UML diagrams least are business analysts, software testers and maintenance engineers. It is sometimes difficult to know whether programmers use the UML diagrams provided by the architects (to "measure" whether the investment of creating the UML diagrams provides any kind of payback). We would therefore like to stress that almost one third (27%) of the programmers use UML diagrams for software maintenance tasks.

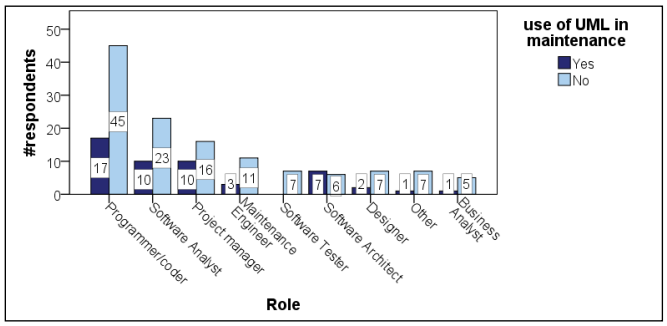

Fig.1. Use of UML diagrams during software maintenance per role

We classified the people who do or do not use UML diagrams during software maintenance by educational level (*Fig.2*). It "seems" that a higher educational level leads to a greater use of UML diagrams (except in the case of PhD students, who may have specialized in topics that are very unlike software modeling). This assumption was made after discarding the results of those groups with low representativeness, due to their low generalizability.

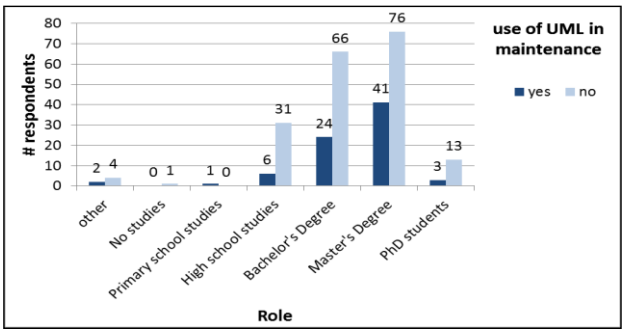

Fig.2. Use of UML diagrams during software maintenance by educational level

The results obtained in answer to the research questions formulated are presented below.

B. RQ1: Is UML Documentation Used to Support Software Maintenance in Industry?

We asked the respondents about what type of documentation they use during software maintenance. The majority use artifacts that are based on textual documentation (66%) and on the source code by itself (70%), but 40% of those surveyed use a graphical notation (26% use only UML,9% another notation), to support the design of the changes related to software maintenance tasks. This means that, when a graphical notation is used, UML is used in 72% of the cases in comparison to other notations. This is more or less the same proportion as in the results obtained by Hutchinson et al. [12] in their survey regarding MDD (Model Driven Development) and the different notations used for it. The percentage of UML use obtained is also similar to that in the results obtained by M. Petre [13] in her study on the use of UML (in general, not only focusing on maintenance). But these two results are contrary to the results obtained by Scanniello et al. [20], which reveal that 75% of the respondents (all from Italian companies) use UML. As stated previously, 20 respondents (9%) use a different graphical notation (like those in the results obtained in [8]). There are also 8 respondents (5%) who use a combination of UML and other graphical notations. This reinforces the results obtained by Hutchinson et al. [12], i.e., the use of different notations is not an unusual practice, because most notations are not selective. The other graphical notations used are the following: BPMN, E/R, SysML, FSP-SPEM, database designs, Archimate, screenshots or domain specific languages, but the majority does not use formalized notations ("boxes and arrows"). Most of them coincide with the languages mentioned in the survey of Hutchinson et al. [12].

Those who do not use any graphical notation but who do employ textual documentation represent 33% of the respondents. 14% do not use any complementary documentation (graphical or textual) to maintain source code, i.e., they only employ the source code as documentation. This is surprising, since it is additional information to the source code and requires an extra investment for its creation. But source code and its comments are the most important artifacts for understanding a system that is to be maintained [14].

From here on we shall use the terms "UML group" for those who stated they have UML diagrams available as part of their maintenance documentation, and "non-UML group" for those who do not have UML diagrams in their documentation.

We asked the UML group how often they do not consult software documentation and work directly with source code (*Table 1*).

Table 1. Frequency of use of source code only

	Never	Some-times	Often	Very often	Always
UML group	4	22	20	2	1
Non-UML group	8	37	43	17	8

Those who indicated they never discard the UML diagrams to work directly with source code form 8% of the UML group. It may be that they do not use source code because of their roles: project managers, software analysts, designers and software architects.

Almost half of the UML group (45%) do not always consult the documentation, and work directly with source code. They justified this way of working ideas as follows:

- There is no documentation in the project.
- Documentation does not always describe the observed behavior of the application.
- It is a useful way to handle defects.
- Source code is always the most reliable documentation.
- When the UML diagrams are created a posteriori (legacy systems) they do not have enough detail.
- UML is not useful when the maintenance task needed is an evolutive task of the system with low impact on the rest of it.

Those who often discard the documentation and work directly with source code (41%) do so when:

- The software maintenance task is small, and it is faster to do it directly in the source (doc. is not needed).
- The system under maintenance is well-known by maintainers. It is recognized that the documentation is not aligned with the source code.
- There is a lack of time.

In the case of the non-UML group (*Table 1*), 7% indicated that they never discard the documentation to work directly with source code.

33% of the non-UML group do not always consult the documentation and work directly with source code. They discard documentation, for the following reasons:

- Sometimes source code is self-explanatory. When the system concerned is very old, the documentation may very well no longer describe the current situation.
- Code sometimes contains more details than documentation.
- Documentation is not necessary for testing.

- There is a lack of tooling for the synchronized updating of source code and documentation.

Those who discard the documentation and directly use source code more often (53%) do so because:

- The documentation documents are very long. The documentation is not properly structured, or not very accessible (parts in emails, for example).
- There is a good deal of knowledge embedded in source code which is not present in the documentation.
- Systems are well-known.
- Documentation might be not updated (only when the customer requires it).
- Documentation does not exist in some cases (especially in legacy systems).
- The maintenance tasks are small or corrective.
- Time pressure exists.

C. RQ2: What Are the Perceived Benefits of Using UML During Software Maintenance?

The UML group was asked why they use UML diagrams. Some possible reasons were presented to them.

The majority of the UML group (55%) believes that the UML based diagram documentation provides added value for software comprehension and defect detection. One recurrent argument (41%) for using UML is the idea that UML based diagram documentation reduces the time needed for software comprehension and defect detection.

Some of them (31%) also think that UML based diagram documentation outperforms the other available standards, diagrams and models. There are also a few (29%) that use it because it has been adopted by their companies, i.e., they are "forced" to do so. A subset of survey respondents (27%) thinks that the UML based diagram documentation reduces maintenance costs.

A minority (8%) use UML because they do not know of any other alternatives. Some of them (8%) also justified the use of UML as follows:

- It is a standard, i.e., it is not ambiguous.
- It is familiar for most developers.
- It is an easy communication model that makes it easier to review the development activity.
- It is easy to understand for technical and non-technical people, because it has different views.

The non-UML group was similarly asked why they do not use UML:

The option chosen most frequently was "I have to use the standards, diagrams and models adopted by my company" (33%). The next reason for not using UML is because the non-UML group prefers working directly with source code (23%). This percentage is double that obtained in the survey of Scanniello et al. [3]. On the other hand, 17% of the non-UML group believes that time spent on UML diagram comprehension is not compensated by the benefits of using UML. This percentage is much lower than the 50% of people surveyed in [15] who excused their non-use of modeling by saying that models require too much effort.

Finally, it is strange that some of those responding do not use UML simply because they are not familiar with it (14%). These data coincide with the results of [8] and [16]. Similarly, 10% of non-UML group believes that the UML based diagram documentation does not add enough value to software comprehension.

A minority (2%) thinks that the standards, diagrams and models used in their companies are better than UML based diagram documentation. Some of them (18%) also argued for the non-use of UML, giving the following reasons:

- It is difficult to manage versions of diagrams. This contradicts the results obtained in [12], in which it was claimed that 50% of companies used versioning tools for modeling (in the context of MDD).
- Legacy systems do not usually have UML diagrams in their documentation.
- UML diagrams are not usually maintained.
- Final customers do not like UML diagrams.
- There is minimal use of documentation, in general.
- When the development starts, the requirements are unclear.
- Diagrams are used for personal purposes but not stored as documentation.

When the questionnaire was created, we took into account that one of the responses to the previous question might be that maintainers use software documentation (containing UML diagrams, or text or other graphical notations) only infrequently. We thought that the effort of consulting the UML/documentation might be great, thanks to that low rate of use. A question about this was therefore added to the questionnaire (*Table 2*).

For the UML group, the effort of consulting the UML diagrams is almost always less than 20% of the total effort made to maintain the system (only 16% of the UML group disagreed with this statement). For the non-UML group, the proportion of those who spent more than 20% of the effort in consulting the documentation is slightly lower. This is because over 1/4 of them use more effort in consulting the documentation, and it may explain why more maintainers in the non-UML group than in the UML group do not use the documentation.

Table 2.Effort of consulting UML diagrams for UML users, and effort of consulting documentation for non UML group

	<10%	**11%-20%**	**21%-30%**	**31%-40%**	**>40%**
UML group	22	19	7	0	1
Non-UML group	46	38	20	3	6

D. RQ3: What Kinds of Companies Use UML Documentation During Software Maintenance?

A "Company" is the final result of several factors, such as organization type, dimension, business domain, type of projects carried out and processes in use. In the survey we attempted to

investigate each of these aspects with a set of focused questions.

We asked those surveyed whether UML use is closely tied to the software development methodology used for software maintenance, to discover if this is indeed the case (*Fig.3*).

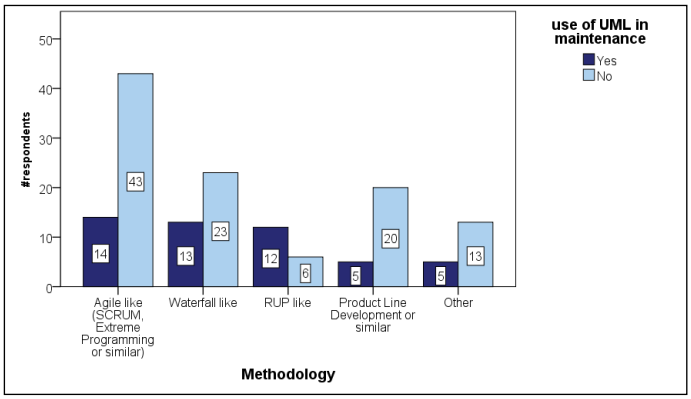

Fig.3. Use of UML diagrams by software development methodology

Only when the development methodology is RUP–like are there more respondents from the UML group than the non-UML group (rate of 2). This proportion was expected, given the particularities of this development process. On the other hand, when following waterfall methodologies or Product Line developments, the proportion of UML users falls drastically (rate of 0.57 and 0.25, respectively).

In the case of other methodologies, namely agile approaches, Prince2, ISO/IEC 29110, MoProSoft, ITIL V3, Polarion, ASAP (SAP), Spiral, there is one UML user per each 3 maintainers.

Some of those taking part also mentioned that they use one or more of their own methodologies, depending on the project.

The modeling tool used to maintain/modify the UML diagrams is an important factor when deciding whether to use an UML based software development process. There are different types of tools with different benefits: licensed tools (which implies an investment but also payback with possible training, customizations, etc.) vs. open tools, or specific tools for modeling in UML (which check the correctness of syntax) or general modeling tools (these are more "accessible").

24% of the UML group does not use a modeling tool: 2% because they manage diagrams on physical paper or blackboards and the diagrams are not digitalized, while the rest (22%) do not manage diagrams because they are not modifiable images (.jpg, .bmp, .pdf, etc.). This is a higher percentage than that obtained in [8], whose results revealed that only 6.4% used modeling tools.

The 73% UML group, which uses tools to modify the UML diagrams, might do so using a single tool, or have more than one available for the same purpose.

It is quite surprising that one of the most frequently-used tools is Microsoft Office Visio (29%), which is non-specific to UML design. It is true that it contains a toolbar for UML design, but it does not check the basic correctness of the UML diagram, i.e., the syntax is not checked. The tool thus allows any element to be connected with another one, or even elements from one diagram to be introduced in another (for example, actors in a class diagram). One reason for the frequent use of this tool might be that a lot of companies work with it for other purposes; employees already have the tool installed, making it easier to use it and to share diagrams.

Enterprise Architect, which is a very comprehensive licensed tool, has the same percentage of use (29%). It contains the option of producing Reverse Engineering (RE) UML diagrams, compared to the next most widely-used tool, which is the open tool StarUML (14%). Others mentioned are (with 2% to 7% each): Rational Rose, ArgoUML, IBM Software Architect, DIA, Visual Studio, Gliffy and UML designer (a plugin for Eclipse).

One of the purposes of using UML diagrams is to improve communication between stakeholders [4], [19]. When a company is geo-distributed, this factor becomes critical. In both cases, fewer companies use UML during maintenance, regardless of their locations. The proportion of companies that use UML diagrams is, surprisingly, a little higher (9% extra) for those companies that are geo-distributed in comparison to those that are co-located (*Table 3*). This could be because co-located teams are more standardized as regards development methods and tools.

It is also worth noting that 83% of the respondents belong to geo-distributed companies. But how often is the maintenance team geo-distributed? Half of the participants work in maintenance teams that are geo-distributed, while the other half are in co-located teams (*Table 4*). We see, however, that the proportion of UML use is slightly lower (9% less) in the case of co-located maintenance teams.

Table 3. Relationship between geo-distribution of companies and use of UML in software maintenance

	Geo-distributed company		**Co-located company**	
	# respon-dents (in total)	**%**	**# respon-dents (in total)**	**%**
Use UML in maintenance	40	28%	21	37%
Do not use UML in maintenance	105	72%	60	63%

Table 4. Relationship between geo-distribution of the maintenance teams and use of UML in software maintenance

	Geo-distributed maintenance team		**Single-site Maintenance team**	
	# respon-dents (in total)	**%**	**# respon-dents (in total)**	**%**
Use UML in maintenance	30	35%	21	26 %
Do not use UML in maintenance	56	65%	60	74 %

We also studied the influence of company size on the use of UML diagrams. We measured the size of companies, using their number of employees. The majority of those responding (66%) belong to very large companies, with more than 250 employees. Large companies (with 50 to 250 employees) use a higher proportion (40%) of UML diagrams than those of other sizes (*Fig.4*). In the rest of the categories, the use is less than 30%.

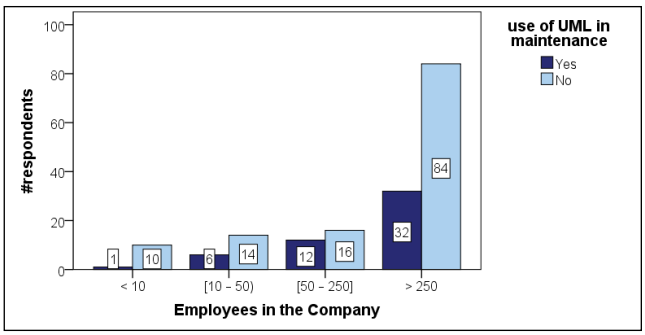

Fig.4. Relationship between size of company and use of UML in software maintenance.

We then looked into the influence of the size of the ICT departments on the use of UML (see *Table 5*). Companies with small ICT departments, i.e., with fewer than 10 employees, use fewer UML diagrams (around 16%) than bigger ICT departments (from 26% to 39%). This was as expected, since one of the main reasons for using UML diagrams is communication between team members; there is expected to be less need for codified design knowledge in small ICT departments.

Table 5.Relationship between size of ICT department and use of UML in software maintenance.

	<10	**[10-50)**	**[50-250]**	**> 250**
Use UML in maintenance	3 (16%)	10 (38%)	12 (39%)	26 (26%)
Do not use UML in maintenance	16 (84%)	16 (62%)	19 (61%)	73 (74%)

Focusing on team size (*Table 6*), we considered small teams to be those with fewer than 3 people, medium to be those with between 5 and 9 people, large to be those with 10 to 49 people, and very large to be those with 50 or more people. The majority of the respondents work in small (44%) or medium (35%) sized teams. Team size does not appear to be an influential factor in the use of UML during software maintenance. It is also important to note that, although the ICT departments are very large, team size does not tend to be correspondingly large. This is why there are more respondents from small teams than from large ones. What is more, small teams use fewer UML diagrams than large ones, because they have facilities for face-to-face meetings, and so need less supporting documentation.

The type of maintenance team was also studied. In most cases (80%) the respondents belong to stable maintenance teams whose objective is to directly develop or maintain software. In the remaining cases (20%), the team was created when needed. Team stability does not seem to be an influential factor in the use of UML; the proportion is the same for both kinds of teams (29%).

Table 6.Relationship between team size and use of UML in software maintenance.

	Small	**Med.**	**Big**	**Very Big**
Use UML in maintenance	25 (33%)	14 (24%)	9 (35%)	3 (50%)
Do not use UML in maintenance	51 (67%)	45 (76%)	17 (65%)	3 (50%)

With regard to the size of the systems maintained, these are classified depending on their number of Lines of Code (LoC). A small system is one with fewer than 10,000 LoC (10% of respondents); a medium system has between 10,000 and 100,000 LoC (38%); a large system might have between 100,000 and 500,000 LoC (33%), while a very large system would have more than 500,000 LoC (19%) [3]. A higher use of UML diagrams was expected in projects that maintain larger systems. One of the reasons put forward for using UML (or models) is that it helps manage large and/or complicated systems. The results obtained show that UML diagrams gain popularity when the team has to maintain a very large system (*Table 7*), but differences are not too great (from 20% to 39%).

Table 7. Relationship between size of system maintained and use of UML in software maintenance

	Small	**Medium**	**Big**	**Very Big**
Use UML in maintenance	4 (29%)	19 (31%)	18 (33%)	10 (39%)
Do not use UML in maintenance	10 (71%)	43 (69%)	37 (67%)	26 (61%)

The type of maintenance most often performed is evolutive, since 82% of the respondents do it frequently (i.e., 60 often, 59 very often or 27 always), followed by corrective tasks. The adaptive maintenance tasks are done less often (33% of the respondents never do so, or do so rarely). These results are similar to those obtained by Souza et al. [14], who state that the most frequently-used maintenance is evolutive.

We studied whether the business sector type of the company influences the use of UML during maintenance. The categories used to classify companies by business sector were the following: Finance (0% of respondents); Telecommunications (2%); Manufacturing (3%); Service Provider (22%); SW Development, Maintenance and Service (69%); or other sectors (4%). Other sectors included education, medicine, or humanitarian companies. If we focus on those respondents who belong to "Software companies", 35% use UML. As expected, there are other sectors not directly dedicated to software development (they create software as a resource for their companies, but their main business focus is on other items) that do not seem to use UML diagrams. In

contrast, a low proportion (13%) of those working in the "Service Provider" sector uses UML diagrams in maintenance, while 40% of those employed in "Manufacturing" companies use the UML diagrams. It is surprising that companies not directly dedicated to software development use UML diagrams (13% of Service Providers, and 40% of Manufacturing Companies surveyed, although these percentages are calculated based on low representativeness).This implies that UML is being used not only in the context of software factories.

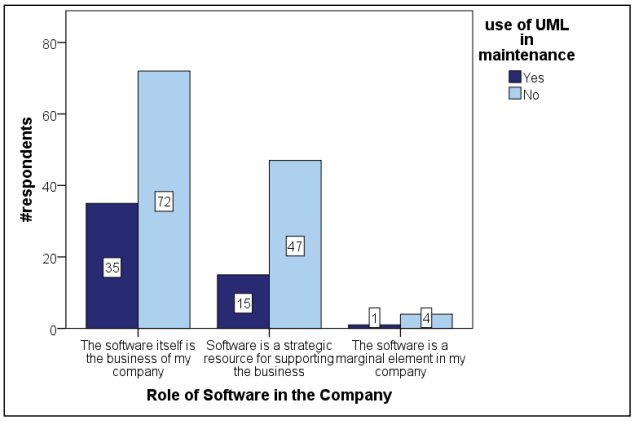

Fig.5. Relationship between role of software in the company and use of UML in software maintenance.

More than half the respondents (66%) are employees of companies in which the company's business is the software itself; i.e., they work in ICT companies (*Fig.5*). Another large percentage of the participants (36%) work for companies in which the software is a strategic resource to support their business, but they focus on producing other kinds of products, or on providing other kinds of services. Only 5 respondents (3%) worked for companies in which software is a marginal element. The percentage of UML usage is higher in the case of those companies in which the software is the main business element (33%), compared to those in which the software is a supporting element for the company's business (24%).

E. RQ4: Which Subset of UML Diagram Types Has Been Demonstrated to Be Most Widely-Used and Effective During Software Maintenance?

The UML diagrams that are most frequently used during software maintenance (*Fig.6*) are class diagrams (61% of UML group), use case diagrams (45%), sequence diagrams (41%) and activity diagrams (33%). These results are similar to those of other surveys ([8], [11], [12], [17])in which class, activity, use case and sequence diagrams are part of the top 4 UML diagrams used. The diagrams used least are the collaboration diagrams (perhaps because they are equivalent to sequence diagrams), composite structure diagrams, interaction view diagrams and timing diagrams (all of which are UML 2.0 diagrams and less well known).

The origin of the UML diagrams is usually the development phase (94%); i.e., they are not created specially during the maintenance, except where UML diagrams are not available from the software development; then they are created specifically during maintenance (3 respondents; 6%). It is also quite surprising that 34 of those replying (44%) said that UML diagrams are available from the development phase, but that they do not use them during software maintenance at all. This might be the result of a divergence between the diagrams and the source code.

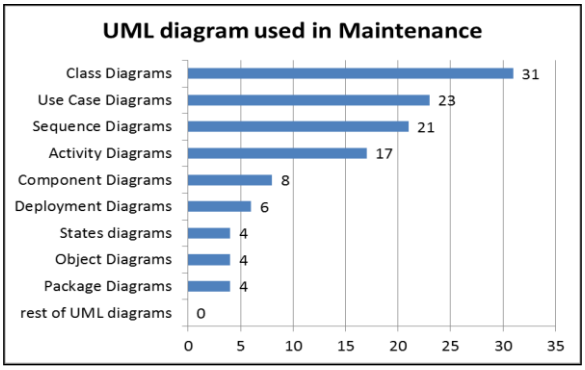

Fig.6. UML diagrams used during software maintenance.

The UML diagrams produced during SW development are summarized in Fig.7. Curiously, some survey respondents (7) considered UML diagrams to be other types of diagrams, such as: Business Process Models, Data models, DDL, Test case, Navigation diagrams, and user interface prototypes. It does not therefore appear to be clear which are UML diagrams and which are not.

When the UML diagrams are created expressly for the maintenance tasks, two approaches might be followed: 1) the diagrams could be human-based diagrams, i.e., they are created manually using a forward design approach, and 2) they could also be machine-based diagrams, i.e., the diagrams are created automatically by software using a Reverse Engineering (RE) technique.

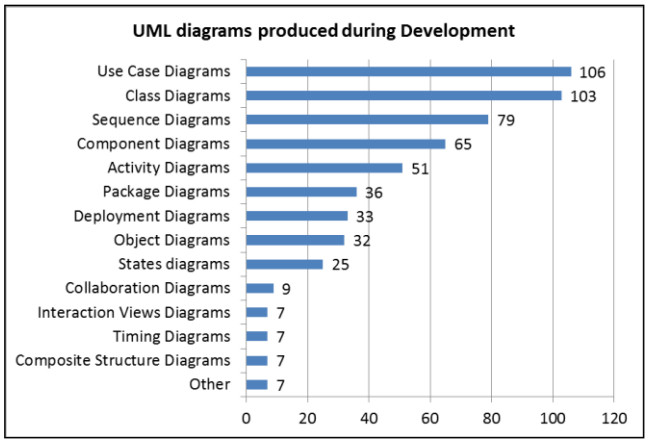

Fig.7. UML diagrams created during software development

If the responses from those who do not know the origin of the UML diagrams (3%) are discarded, the majority of the UML groups use human-based diagrams (81%). Of the rest, those who use UML diagrams generated using any tool with a RE technique, 3% use pure RE diagrams (completely automatic UML diagrams). 19% employ RE UML diagrams reviewed by humans (which could be considered as semiautomatic diagrams). In relation to this, the results of [18] show a

tendency toward better results being obtained when using UML diagrams (class diagrams, specifically) that were hand-made during the design phase. The results from [18] also revealed that maintainers using RE diagrams experienced more difficulties when reading the diagrams. Most companies surveyed therefore use the "more understandable" UML diagrams. Maintainers do not always employ the available documentation and work directly with the source code; even if the documentation is available, it is not used.

IV. Threats to Validity

We shall now analyze the main potential threats to the validity of the survey presented here:

Internal validity: the main issues affecting the internal validity of our study concern the framing and sampling of the participants. Our recruitment strategy could have incurred a possible selection bias (for example, a high probability of profile similarity among the respondents). However, we believe that the issues analyzed are not affected by this threat; the sample size is large and there is variety in the roles and nationalities of respondents. Another threat derives from the channel used to survey maintainers; the questions may have been answered by respondents who did not have the knowledge required to do so. We attempted to address this issue when we defined the protocol of the survey: we explicitly required the survey to be filled in by ICT professionals (the target roles were specified) involved in the maintenanceof software systems (the definition of maintenance was provided). Another negative factor could have been the difficulty involved in understanding the questions (e.g., ambiguous, unclear, not well-formulated), and the respondents' motivations might also have affected the answers and thus the survey results. In web-based surveys the sampling procedure makes it possible to select duplicate units; one person might answer the survey more than once. We addressed this threat by using a system consisting of a single link per person. The reader may also object that the companies within our industrial network might also have influenced the internal validity, and that several people from the same company may have answered the survey, thus biasing the results.

External validity: To interpret the results we obtained correctly, it should be borne in mind that, although the demographics of our sample are fairly diverse, generalizing our results to the entire population may not be appropriate. In our survey, the companies belonged to a variety of domains and covered different company sizes in various countries throughout the world; we cannot be certain that our sample is representative of the ICT industry in general, however. These threats are always present in industrial surveys.

V. Conclusions and Future Work

This paper reports the findings of a survey on the use of UML in software maintenance to which178 ICT professionals responded. The main findings, grouped by RQs, are:

- RQ1: Is UML documentation used to support software maintenance in industry? 59% of those surveyed use a graphical notation (43% UML; 16% another notation) as a complement in trying to understand the system that will be maintained. In contrast, 28% of the respondents use only source code; they consider that source code and its comments are the most important artifacts in understanding the system to be maintained. It is quite surprising that maintainers do not always use the UML diagrams that are available from the development phase. This might be due to problems of a lack of synchronization caused by non-updated diagrams.

- RQ2: What are the perceived benefits of using UML during software maintenance?The main reasons for using UML are that less time is needed for a better understanding of the system under maintenance; this improves defect detection. Reasons given for not using UML are that maintainers follow other standards provided by their companies or that they prefer to work directly with source code. Moreover, when UML diagrams are available as part of the documentation, it takes less effort to consult them. That might explain why more maintainers do not use the documentation in the group whose documentation contains UML than in the group whose documentation does not.

- RQ3: What kinds of companies use UML documentation in software maintenance? The size of the maintenance team appears to influence the use of UML. Larger teams use UML diagrams more frequently, proportionately, perhaps because of the improvement to the understanding of the system provided by UML diagrams and due to the need to share/communicate knowledge in this kind of teams. The size of the system being maintained also seems to be an influential factor. The results obtained show that UML diagrams are extremely popular when the team has to maintain a very big system. This seems to be logical, since one of the reasons put forward for using UML (or models) is that it helps manage large and/or complicated systems.

Regarding additional results for characteristics of companies, we should note that geo-distributed maintenance teams are common (50% of the cases), although this is not influential in UML use. Moreover, the most common type of maintenance is evolutive. It is also noteworthy that maintenance tasks seem to be carried out by those who already have experience in ICT, i.e., they have worked in development before maintenance (first they learn to create it, and then they learn to change it).

Moreover, and surprisingly, Visio is the most commonly-used tool for UML modeling, although it is not specific to UML. It is followed by Enterprise Architect (licensed tool) and StarUML (open source tool).

- RQ4: Which subset of UML diagram types has been demonstrated to be most widely-used and effective during software maintenance?As expected, the UML diagrams that are used most frequently during software maintenance are class diagrams, use case diagrams, sequence diagrams and activity diagrams.

UML diagrams are used most frequently by architects (54%), SW analysts (30%) and managers (39%). But 40% of programmers use UML diagrams for software maintenance tasks, implying that the investment made by architects when creating UML diagrams has a payback: improvement of

maintainer understanding. We noted, however, that there are some (1%) maintainers who do not have diagrams available from the development phase. In other cases, there are some maintainers (23%) for whom diagrams are available, but never used. This study also revealed that the educational level seems to be an influential factor as regards the use or not of UML. Apparently, the higher the education level, the greater the use of UML diagrams.

Additionally, and in relation to UML documentation, a majorityuseUML diagrams that are human-based diagrams (81%). The rest use UML diagrams generated using RE with a tool: 3% use pure RE diagrams, while 19% use RE UML diagrams reviewed by humans.

The results of this survey might be beneficial for helping companies see how to invest in making the systems being maintained easier to understand. These results give us grounds to encourage software developers, albeit with caution, to develop UML diagrams in the early stages of software development. That would facilitate future maintenance tasks, encouraging maintainers to keep diagrams updated.

These findings are a first approach to discovering the contexts in which companies use UML during software maintenance; we plan to investigate this topic in greater depth. This future work might take the form of a survey that includes open questions or interviews. We also wish to extend our investigation to involve industries that do not belong to our industrial contact networks.

ACKNOWLEDGMENTS

This work has been funded by the following projects: GEODAS-BC (Ministerio de Economía y Competitividad and Fondo Europeo de Desarrollo Regional FEDER, TIN2012-37493-C03-01) and IMPACTUM (Consejería de Educación, Ciencia y Cultura de la Junta de Comunidades de Castilla La Mancha, y Fondo Europeo de Desarrollo Regional FEDER, PEII11-0330-4414).

REFERENCES

[1] OMG, "The Unified Modeling Language. Documents associated with UML version 2.3," 2010. [Online]. Available: http://www.omg.org/spec/UML/2.3.

[2] W. J. Dzidek, E. Arisholm, and L. C. Briand, "A realistic empirical evaluation of the costs and benefits of UML in software maintenance," *IEEE TSE*, vol. 34, no. 3, pp. 407–432, 2008.

[3] G. Scanniello, C. Gravino, and G. Tortora, "Investigating the Role of UML in the Software Modeling and Maintenance - A Preliminary Industrial Survey," in *Proceedings of the ICEIS'2010*, Funchal, Madeira, Portugal, 2010, vol. 3, pp. 141–148.

[4] A. M. Fernández-Sáez, M. R. V. Chaudron, and M. Genero, "Exploring Costs and Benefits of Using UML on Maintenance: Preliminary Findings of a Case Study in a Large IT Department," in *Proceedings of EESSMoD'2013, celebrated within MODELS'2011,* 2013, pp. 33–42.

[5] A. Pinsonneault and K. L. Kraemer, "Survey research methodology in management information systems: an assessment," *Journal of Management Information Systems - Special section: Strategic and competitive information systems*, vol. 10, no. 2, pp. 75–105, 1993.

[6] B. A. Kitchenham and S. L. Pfleeger, "Principles of survey research part 2: designing a survey," *ACM SIGSOFT Software Engineering Notes*, vol. 27, no. 1, pp. 18–20, 2002.

[7] E. B. Swanson, "The dimensions of maintenance," in *Proceedings of ICSE 1976*, San Francisco, California, United States, 1976, pp. 492–497.

[8] L. T. W. Agner, I. W. Soares, P. C. Stadzisz, and J. M. Simão, "A Brazilian survey on UML and model-driven practices for embedded software development," *Journal of Systems and Software*, vol. 86, no. 4, pp. 997–1005, 2013.

[9] F. Tomassetti, M. Torchiano, A. Tiso, F. Ricca, and G. Reggio, "Maturity of software modelling and model driven engineering: A survey in the Italian industry," in in *Proceedings ofEASE 2012*, 2012, pp. 91–100.

[10] SurveyMonkey, "http://www.surveymonkey.com." .

[11] M. Grossman, J. E. Aronson, and R. V. McCarthy, "Does UML make the grade? Insights from the software development community," *Information and Software Technology*, vol. 47, no. 6, pp. 383–397, Apr. 2005.

[12] J. Hutchinson, J. Whittle, and M. Rouncefield, "Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure," *Science of Computer Programming*.

[13] M. Petre, "UML in practice," in *Proceedings of ICSE'2013*, San Francisco, USA, 2013, pp. 722–731.

[14] S. C. B. de Souza, N. Anquetil, and K. M. de Oliveira, "A study of the documentation essential to software maintenance," in *Proceedings SIGDOC'2005*, 2005, pp. 68–75.

[15] M. Torchiano, F. Tomassetti, F. Ricca, A. Tiso, and G. Reggio, "Preliminary Findings from a Survey on the MD State of the Practice," in *Proceedings of ESEM 2011*, 2011, pp. 372–375.

[16] A. Nugroho and M. R. V. Chaudron, "A survey into the rigor of UML use and its perceived impact on quality and productivity," in *Proceedings of ESEM 2008*, 2008, pp. 90–99.

[17] B. Dobing and J. Parsons, "How UML is used," *Communications of the ACM*, vol. 49, no. 5, pp. 109–113, 2006.

[18] A. M. Fernández-Sáez, M. Genero, M. R. V. Chaudron, D. Caivano, and I. Ramos, "Are Forward Designed or Reverse-Engineered UML diagrams more helpful for code maintenance?: A family of experiments," *Information and Software Technology*, vol. 57, no. 1, pp. 644–663, 2015.

[19] E. Tryggeseth, "Report from an Experiment: Impact of Documentation on Maintenance," *Journal of Empirical Software Engineering*, vol. 2, no. 2, pp. 201–207, 1997.