CHAPTER 2. STATE-OF-THE-ART

Fernández-Sáez, A. M., Genero, M., and Chaudron, M. R. V. (2013). 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(1) pp. 1119-1142 (**JCR Index 2013 = 1.522**).

ABSTRACT

Context: The Unified Modelling Language (UML) has, after ten years, become established as the de facto standard for the modelling of object-oriented software systems. It is therefore relevant to investigate whether its use is important as regards the costs involved in its implantation in industry being worthwhile.

Method: We have carried out a systematic mapping study to collect the empirical studies published in order to discover "What is the current existing empirical evidence with regard to the use of UML diagrams in source code maintenance and the maintenance of the UML diagrams themselves?

Results: We found 38 papers, which contained 63 experiments and 3 case studies. *Conclusion*: Although there is common belief that the use of UML is beneficial for source code maintenance, since the quality of the modifications is greater when UML diagrams are available, only 3 papers concerning this issue have been published. Most research (60 empirical studies) concerns the maintainability and comprehensability of the UML diagrams themselves which form part of the system's documentation, since it is assumed that they may influence source code maintainability, although this has not been empirically validated. Moreover, the generalizability of the majority of the experiments is questionable given the material, tasks and subjects used. There is thus a need for more experiments and case studies to be performed in industrial contexts, i.e., with real systems and using maintenance tasks conducted by practitioners under real conditions that truly show the utility of UML diagrams in maintaining code, and that the fact that a diagram is more comprehensible or modifiable influences the maintainability of the code itself. This utility should also be studied from the viewpoint of cost and productivity, and the consistent and simultaneous maintenance of diagrams and code must also be considered in future empirical studies.

Keywords: UML, empirical studies, software maintenance, Systematic Mapping Study, Systematic Literature Review

2.1. INTRODUCTION

UML was first introduced in 1997, and became a de facto standard for the modelling of object-oriented software systems in 2000 (OMG, 1997). It subsequently evolved and the latest version appeared in 2009 (UML 2.3) (OMG, 2010).Owing to the increasing complexity of software projects at the present time, the UML has emerged as a tool which is being used to increase the understanding between customers and developers (in the analysis phase). It is also being employed to improve the communication among team members (Nugroho and Chaudron, 2009) and to broaden the understanding of how software works (in both the development and maintenance phases).

Despite all this, any type of investment must be justified from an economic point of view, in the sense that there should be a payback at a later phase. In the context of software projects, therefore, investment in modelling should be justified by benefits that can be gained later, during software development or maintenance. Such benefits

might include improved productivity and better product quality. The existence of these potential advantages is one of the main reasons for investigating whether the use of the UML can generate important differences that make the costs worthwhile. This is particularly true in the context of software maintenance, which consumes a large part of software development resources. Maintenance typically accounts for 40 to 80 percentage of software costs (Glass, 2002; Pressman, 2005).

More than fifteen years have passed since 1997, when the UML was first introduced as a modelling language to describe object-oriented software systems. A comprehensive review of the empirical literature on software engineering is an important step towards its use in maintenance. That being so, it would be useful for the software industry to know what empirical evidence exists as regards the use of the UML in the maintenance of source code and the maintenance of the UML diagrams themselves. With this purpose in mind, we decided to perform a review of the literature related to this issue in order to answer our main research question:

What is the current existing empirical evidence with regard to the use of UML diagrams in source code maintenance and the maintenance of the UML diagrams themselves?

The scientific literature found differentiated several types of systematic reviews (Petticrew and Roberts, 2006), including the following:

- Conventional systematic reviews (Petticrew and Roberts, 2006), which aggregated results concerning the effectiveness of a treatment, intervention, or technology, and were related to specific research questions, and
- Mapping studies (Arksey and O'Malley, 2005) whose aim was to identify all research related to a specific topic, i.e. to answer broader questions related to trends in research. Typical questions are exploratory.

This paper aims to present a systematic review of papers dealing with the use of UML diagrams in source code maintenance and the maintenance of the UML diagrams themselves. This work is classified as a secondary study since it is a review of primary studies. A proper systematic review of the literature follows a rigorous and systematic approach, like that described by Kitchenham and Charters in (Kitchenham and Charters, 2007). This approach was therefore used as a basis to perform a systematic mapping study, owing to the need to adopt systematic approaches towards assessing and aggregating research outcomes in order to provide a balanced and objective summary of research evidence for this particular topic. Our goal was to collect evidence that could be used to guide research and practice, and we therefore consider this systematic mapping study to be part of the evidence based software engineering effort (Kitchenham and Charters, 2007).

With regard to our main research question, it is important to note that, on the one hand, most companies only maintain the source code of a system without updating the diagrams which represent it (Forward et al., 2010). This may influence the subsequent maintenance of the same system, which might be misunderstood as a result of inconsistencies. The experience of several researchers belonging to the FaST-RE

research group (Leiden University), headed by Michel R. V. Chaudron, which has a long tradition of collaborating with industrial partners, reflected that the lack of maintenance of diagrams can occur for several reasons such as time constraints, a low level of comprehension of diagrams, or a low modifiability of diagrams.

On the other hand, software development itself is becoming more model-centric (Mohagheghi et al., 2009). Both the OMG Model Driven Architecture (OMG, 2003) and the recent growth of the Model-Driven Development (MDD) software engineering paradigm (Atkinson and Kühne, 2003) emphasize the role of modelling in the development (and hence in the maintenance) of software systems. MDD treats software development as a set of transformations between successive models, from requirements to analysis, to design, to implementation, and to deployment (Thomas, 2004). MDD's defining characteristic is that the primary focus and products of software development/maintenance are models, rather than computer programs.

These facts also led us to focus on the importance of investigating the maintenance of the models/diagrams themselves, and of attempting to discover whether the diagrams are understandable and modifiable to an extent that would allow maintainers to perform the changes that need to be made to them at the same time as they are maintaining the source code.

At this point, we shall define what maintenance is considered to be in this paper. The software production process can be broken down into various phases. The most commonly-defined phases are the following: analysis and requirements definition, design, implementation, testing and installation, and operation and maintenance (Priestley and Utt, 2000).

Maintenance is defined as the modification of a software product after it has been delivered (to users or customers) in order to correct faults, improve performance or other attributes, add new functionalities, or adapt it to a changing environment (IEEE, 1993).

As is common in the maintenance literature (Genero et al., 2001), we considered two of the major types of tasks included in maintenance:

- Understanding/comprehending the software artifact: in order to modify a program, programmers need to understand its functionality and requirements, its internal structure and its operating requirements. This is necessary if the impact of changes is to be understood.
- Modification of the software artifact: in order to incorporate the necessary changes, a maintenance engineer should create, modify and verify data structures, logic processes, interfaces and documentation. Programmers should have an in-depth knowledge of the impact on the system of the changes being made in order to avoid possible side effects.

The aforementioned tasks used as part of the maintenance and the maintainability sub-characteristics proposed in the ISO 25000 (ISO/IEC, 2008) were additionally used as a basis to focus our study on modifiability as part of maintenance tasks. Moreover, although understandability is not considered to be a maintainability sub-characteristic

in the ISO 25000, we consider it to be part of maintenance since a considerable amount of works judge understandability to be a factor that influences maintainability (Briand et al., 2001a; Deligiannis et al., 2002; Genero et al., 2007; Harrison et al., 2000). A software artefact must be well-understood before any changes can be made to it. We also took misinterpretation as being a factor that influences the understandability of a system and thus its maintainability.

The remainder of this paper is organized as follows. Section 2.2 presents a brief discussion of related work. This is followed by the explanation of each step of the systematic mapping study process. The explanation of the steps involved in *planning* and *conducting* the study can be found in Sections 2.3 and 2.4, respectively. The *reporting* results step and data synthesis of the systematic mapping study are presented in Section 2.5. A discussion of the results is presented in Section2. 6, along with the implications of these findings. Section 2.7 presents the threats to the validity of this systematic mapping study. Finally, in Section 2.8, the conclusions reached are set out and future research possibilities are discussed.

2.2. RELATED WORK

To the best of our knowledge only one research study which presents a literature review on the effect that the use of the UML by developers has on the design and maintenance of object-oriented software (Dzidek, 2008), and this review is different from that performed here. On the one hand, it is worth noting that the process carried out by Dzidek (Dzidek, 2008) is not strictly a systematic literature review or a systematic mapping study. The study relies on the terms that a systematic literature review or systematic mapping study provides, but some activities are missing. In the *planning* step of the review there are no details about the creation of a comprehensive review protocol. Moreover, the research study limits the search for documents to a number of journals and conferences that the author knows of and considers relevant (i.e., a manual search was performed). The problem with this is that there are also some important conferences on the subject, such as the International Conference on Software Maintenance (ICSM) or the European Conference on Software Maintenance and Reengineering (CSMR), among others, which were not taken into account by the author. Another difference is the search period, since in our study the period under study is broader (from 1997 until 2010) than that in (Dzidek, 2008), which covers the period until 2006. In addition, the research questions were different. Dzidek focuses on efforts to find documents that are relevant to the state-of-the-art of the use of the UML in industry, and also seeks to discover the influence of using supporting tools. In contrast, although the systematic mapping study presented in this paper also aims to obtain the state-of-the-art of empirical studies related to the use of the UML in maintenance tasks, it fixes its attention on how rigorously such empirical studies have been performed (variables, diagrams used, threats to validity, etc.) rather than concentrating on the tools used. The effect of all these differences is that Dzidek obtained 23 primary studies, which is half the number of primary studies found in the systematic mapping study presented in this paper, and only 10 papers are selected as primary studies in both pieces of work.

Another study, that of Budgen (Budgen et al., 2011a), does report a systematic literature review of empirical studies related to the UML, but its focus is different from our systematic mapping study. It aims to discover empirical studies concerning the use of the UML in general, in addition to those concerning some of the properties of the UML. Our focus, however, is specifically on the use of UML diagrams in software maintenance. We do not consider those papers that deal with the properties of the UML as a language, since we have concentrated on factors of UML diagrams when they are used in software maintenance. The aforementioned study, moreover, took as its main task an investigation into the methods and tools used for UML notations or extensions (such as the Object Constraint Language (OCL)). In addition, the papers in it were only those published until 2008, and our study period is therefore broader. It is also important to highlight that its data extraction form concentrates mainly on obtaining data related to how the empirical study was carried out, and on the type of subjects and tasks. The environments in which the studies were carried out, the kind of systems used, or the origin of the diagrams are not considered.

Genero et al. (Genero et al., 2011) present a systematic literature review on the quality of UML diagrams, but it is not focused exclusively on maintenance, as the systematic mapping study reported on in this paper is (less than 20% of the primary studies were related to maintainability of the UML diagrams). It is worth mentioning that although they did not focus solely on empirical evidence, 30% of their primary studies were empirical (24% of the total were experiments, 5% case studies, and 1% surveys).

Despite the fact that UML is widely used in practice, little is known about how UML is actually used. A survey on the use of UML presented by Dobing & Parsons in (Dobing and Parsons, 2006) describes which UML notations (diagram types) are commonly used. These authors analyze the use of each diagram from different points of view, i.e., how are they used for client verification, for programmers' specifications or for maintenance documentation purposes. This last category, i.e., maintenance documentation, is that which is most closely related to the purpose of the systematic mapping study presented herein. The authors stated that class, sequence, and use case diagrams are most often used in practice.

2.3. PLANNING

Planning includes dividing the workload amongst the researchers and determining how the researchers will interact and conduct the review; it also encompasses the development of the review protocol itself. The *planning* step is concerned with developing the protocol that prescribes a controlled procedure for *conducting* the review. Our protocol included objectives, research questions, a search strategy, and inclusion/exclusion criteria (as part of the selection strategy), along with a data extraction form and the quality assessment criteria. The protocol was revised and refined in iterations after the execution of each of the respective activities in the review.

Our main objective is to gather empirical evidence in order to discover whether software maintainers perform maintenance tasks better (in terms of less time and fewer defects) when a UML diagram is available, or whether the use of UML diagrams does not decrease maintainers' productivity or quality. We focus on maintenance in general, independently of whether the maintenance is performed solely on the code, or also on the diagrams, or only on the diagrams and then translated to code. This objective allowed us to derive a series of questions that we hoped to answer with the results of our research:

RQ1: Which diagrams are most frequently used in studies concerning the maintenance of UML diagrams or the maintenance of source code when using UML diagrams?

RQ2: Which dependent variables are investigated in the empirical studies?/ How are they measured?

RQ3: What is the state-of-the-art in empirical studies concerning the maintenance of UML diagrams or the maintenance of source code when using UML diagrams?

RQ4: Which of the factors studied influence the maintainability of a system (source code and diagrams)?

We aim to gather the existing empirical evidence within the area of the maintenance of the UML diagrams or their use in maintenance tasks. In particular, we wish to take into account empirical research on the topic. The latter is particularly important, since it provides information about what we actually know in terms of evidence.

Based on our research questions, we selected the major search terms, which are "UML", "Maintenance" and "Empirical". The alternative spellings and synonyms of, or terms related to, the major terms are denominated as alternative terms (and are shown in Table 2.1). The search terms were constructed using the following steps (Brereton et al., 2007):

1. Define major terms.
2. Identify alternative spellings and synonyms of, or terms related to, major terms.
3. Check the keywords in any relevant papers we already have.
4. Use the Boolean OR to incorporate alternative spellings, synonyms or related terms.
5. Use the Boolean AND to link the major terms.

Only "Unified Modelling Language" was considered to be synonymous with UML, rather than adding the name of each UML diagram. This was because we aimed to cover all of the thirteen diagrams that the UML includes.

As explained in the introduction, for terms related to maintenance, we took all the maintainability sub-characteristics proposed in the ISO 25000 (ISO/IEC, 2008). Although understandability is not considered to be a maintainability sub-characteristic in the ISO 25000, we included terms related to understandability since a considerable amount of works judge understandability to be a factor that influences maintainability (Briand et al., 2001a; Deligiannis et al., 2002; Genero et al., 2007; Harrison et al.,

2000). A software artefact must be well-understood before any changes can be made to it. So, we also took misinterpretation to be one of the factors that influence the understandability of a system and thus its maintainability. This being the case, we added this term to the search string.

Table 2.1. Search string.

Major terms	Alternative terms
UML	Unified Modelling Language
Maintenance	Maintainability, Modularity, Reusability, Analyzability, Changeability, Evolution, Evolvability, Modification, Stability, Testability, Comprehensibility, Comprehension, Understandability, Understanding, Misinterpretation
Empirical	Experiment, Survey, Case study, Action research

We performed and automated searches in 6 digital libraries rather than performing a manual search based on the following assumptions (although subsequently published literature (Kitchenham et al., 2010) contradicts one of these assumptions): 1) it saves times during the search; 2) all the sources in digital libraries are correctly indexed, so all the available sources, i.e., conferences and journals, will be taken into account, thus contributing towards improving the completeness of the results; and 3) if the search string is well constructed and it is sufficiently robust, all available research will be found.

The complete search strategy is summarized in Table 2.2.

Table 2.2. Summary of search strategy.

Databases searched	• SCOPUS database • Science@Direct with the subject Computer Science • Wiley InterScience with the subject of Computer Science • IEEExplore • ACM Digital Library • SPRINGER database
Target items	• Journal papers • Workshop papers • Conference papers
Search applied to	Abstract – when this was not possible we searched in the full text
Language	Papers written in English
Publication period	From January 1997 to January 2010 (inclusive)

The papers that were included were those that presented any kind of empirical study dealing with the use of the UML in maintenance-related tasks, which had been written in English and which were published between 1997 and January 2010. As the UML was adopted by OMG in 1997 (OMG, 1997), it made no sense to search before that period.

The following papers were excluded: pure discussion and opinion papers, studies available only in the form of abstracts or PowerPoint presentations, duplicates (for example, the same paper included in more than one database or in more than one

journal), research focusing on issues other than maintenance processes using the UML and their empirical validation, or where the major terms were only mentioned as a general introductory term in the paper's abstract. Papers were also excluded if they dealt with extensions to the UML, because our interest lay in the UML itself, in the form specified by the OMG.

A summary of the selection strategy is shown in Table 2.3.

Table 2.3. Summary of selection strategy.

Inclusion criteria	• Only English • Date of publication: from January 1997 to March 2010 • Published and refereed works • Terms satisfying the search string
Exclusion criteria for titles and abstracts	• Pure discussion and opinion papers, studies available only in the form of abstracts or PowerPoint presentations • Where the UML or maintenance are mentioned only as general introductory terms in the paper's abstract and an approach or another type of proposal is among the paper's contributions
Exclusion criteria for full text	• Papers that deal with UML extensions • Papers that do not contain results of empirical studies • Papers that are a summary of a workshop

A template (Table 2.4) for data extraction was produced to ease the activity of synthesizing the data gathered, inspired by the work of (Šmite et al., 2010). Each of the papers was classified into several categories, signifying that the template has two parts, the first of which is related to the metadata of the paper (title, author and name of publication) and the second one of which is related to the classification of the paper according to the following categories:

1. **Year of publication**: The year in which the paper was published. This field is not related to any RQ, but it contributes additional results.
2. **Type of publication**: This could be a journal, a conference or a workshop. This field is not related to any RQ, but it contributes additional results.
3. **Empirical methods**: This could be a case study, survey, experiment, or action research. This field is related to RQ3.
4. **Contexts**: This could be a laboratory or an industrial context. This field is related to RQ3.
5. **Number of subjects**: This represents the sample size, or number of subjects involved in the empirical study. This field is related to RQ3.
6. **Type of subjects**: This could be students, professors, and professionals. This field is related to RQ3.
7. **Dependent variables and their measures**: the dependent variables selected and the measures used to measure them. This field is related to RQ2.
8. **Independent variables**: the independent variables selected in the study, i.e., what the treatments compared were. This field is related to RQ3.

Table 2.4. Data extraction form.

Title								
Authors								
Source (name of the journal/conference)								
Year of publication								
Type of publication	**Conference**		**Journal**		**Workshop**			
Empirical method	**Survey**	**Case study**	**Action Research**	**Experiment**				
Context	**Industry**		**Laboratory**					
Number of subjects								
Type of subjects	**Students**		**Academic staff**		**Professionals**			
Dependent variables/ Measures								
Independent variable								
Tasks (number and duration)								
Available diagrams								
Object to maintain	**Code**	**Diagrams**	**Code + Diagrams**					
Type of system	**Synthetic**		**Real**					
Origin of diagrams	**Reverse Engineering**		**Development process**					
Summary/ Comments								

9. **Tasks**: the type of task performed during the empirical studies (test of the understandability of a diagram through a questionnaire, modification tasks, etc.) and its duration (expressed in minutes). This field is related to RQ2.
10. **Available diagrams**: In the original version of the UML submitted in 1997, there were 9 different diagrams with which to model systems from different viewpoints. In UML 2.0 there are 4 new diagrams, making a total of 13. One of these, the communication diagram, has a different name to that of the original UML collaboration diagram. We use the name from the original version, as it is seen more frequently. This field is related to RQ1.
11. **Objects to maintain**: the empirical study can deal with maintenance tasks in the code, in the code and diagrams or in the diagrams only. This field is related to RQ3.

12. **Type of system**: the diagrams could represent a real system or a prototype created specifically for the experiment, which we have called a synthetic system. This field is related to RQ3.
13. **Origin of diagrams**: this could be reverse engineering or a development process. This field is not related to any RQ, but it contributes additional results.
14. **Summary/Comments**: a brief description of what is done in the paper, and which factors were studied. This field is related to RQ4.

As quality criteria for the primary study selection we decided to include those papers that have been published in refereed sources and that also contain empirical data. In addition, as is suggested in (Kitchenham and Charters, 2007), a quality checklist was defined for data synthesis and analysis. As the research field is still immature, and since there are no other review papers on the same topic, we did not want to exclude papers. The quality assessment was to be performed once the primary studies had been selected, the purpose being to assess the rigor of each empirical study. We planned to verify whether or not the publications did indeed either mention or discuss issues related to each of the quality metrics.

The criteria used for quality assessment were based on 8 questions (see Table 2.5) which were extracted from previous work, such as the quality criteria presented by Dybå and Dingsøyr (Dybå and Dingsøyr, 2008a, 2008b), who based their quality assessment criteria on the Critical Appraisal Skills Programme (CASP) (CASP UK, 2008) and principles of good practice for conducting empirical research in software engineering (Kitchenham et al., 2002). Only a few minor changes were made in order to customize the detailed sub-criteria presented in Appendix B of (Dybå and Dingsøyr, 2008a) to our study. A summary of the quality assessment criteria is presented in Table 2.5, along with the way in which each criterion was scored.

2.4. CONDUCTING THE REVIEW

Before presenting the *conducting* step, we wish to clarify that in the remainder of this document, the term *paper* is used to refer to the articles published in conferences, journals or workshops that have been retrieved and are analysed in this literature review. The term *primary study* (or *empirical study*) is used to refer to an experiment or another type of empirical study reported in each paper. Both original studies and replications are counted as primary studies. As a paper may content more than one primary study, the total number of primary studies is greater than the total number of papers.

The *conducting* step includes data retrieval, study selection, data extraction, and data synthesis. In this section, the execution of these activities, performed according the protocol defined above, is explained.

Three researchers were involved in the review, which took around 6 months to complete, and a schedule of which is shown in Table 2.6. This illustrates the *planning*, *conducting* and *reporting* steps on a time scale, along with the outcomes obtained as part of each step.

Table 2.5. Quality checklist.

Quality criteria	Quality metrics (Max=40points)
1. Regarding *Aims* and *Objectives*	**Total: 5 points**
1.1. Is there a clear statement of the aims of the research?	If there is a statement with the aims/objectives of the research +1
1.2. Is there a rationale for why the study was undertaken?	If there is an explanation of the reason for undertaking the research +1
1.3. Do the authors state research questions?	If the RQ are presented +1
1.4. Do the authors state hypotheses and their underlying theories?	If the hypotheses are presented +1 If the hypotheses are explained +1
2. Regarding the *Context*	**Total: 5 points**
2.1. Do the authors describe the sample and experimental units (=experimental materials and participants as individuals or teams)?	If the materials are presented +1 If the participants are presented +1
2.2. Was the recruitment strategy appropriate to the aims of the research?	If the recruitment strategy is explained +1
2.3. Do the authors explain how experimental units were defined and selected?	If the selection of materials is presented +1 If the selection of subjects is presented +1
3. Regarding the *Design of the Experiment*	**Total: 2 points**
3.1. Has the researcher justified the research design (e.g., have the authors discussed how they decided which methods to use - blocking, within or between-subject design; do treatments have levels)?	If the design of the experiment is justified +1
3.2. Do the authors define/describe all treatments and all controls?	If there is an explanation of the treatments +1
4. Regarding *Control Group*	**Total: 1 point**
4.1. Was there a control group with which to compare treatments?	If there is a control group +1
5. Regarding *Data Collection*	**Total: 5 points**
5.1. Are all measures clearly defined (e.g., scale, unit, counting rules)?	If the measures are defined +1
5.2. Is it clear how data was collected (e.g., semi-structured interviews, focus group, etc.)?	If there is an explanation of how the data were collected +1
5.3. Is the form of the data clear (e.g., tape recording, video material, notes, etc.)?	If there is an explanation of the kind of data +1
5.4. Are the tasks clearly defined (multiple choice, open questions, etc.)?	If there is an explanation of the tasks +1
5.5. Are quality control methods used to ensure consistency, completeness and accuracy of collected data?	If the control methods are explained +1
6. Regarding *Data Analysis Procedures*	**Total: 10 points**
6.1. Do the authors justify their choice/describe the procedures/provide references to descriptions of the procedures?	If there is a justification of the choice +1 If there is description of procedures +1 If there are references to procedures +1
6.2. Do the authors report significance levels, effect sizes and power of tests?	If there is a significance level +1 If there is an effect size +1

Quality criteria	Quality metrics (Max=40points)
	If there is a power of test +1
6.3. If outliers are mentioned and excluded from the analysis, is this justified?	If there is an explanation of outliers +1
6.4. Has sufficient data been presented to support the findings?	If there is sufficient data +1
6.5. Do the authors report or provide references to raw data and/or descriptive statistics?	If there is a link to raw data +1 If there are descriptive statistics +1
7. Regarding *Threats to Validity / Bias*	**Total: 7 points**
7.1. Has the relationship between researchers and participants been adequately considered?	If the relationship has been considered +1
7.2. If the authors were the developers of some or all of the treatments, do the authors discuss the implications of this anywhere in the paper?	If the implications were discussed +1
7.3. Was there random allocation to treatments?	If allocation is random +1
7.4. Was training and conduct equivalent for all treatment groups?	If training is equivalent +1 If conduct is equivalent +1
7.5. Was there allocation concealment, i.e. did the researchers know to which treatment each subject was assigned?	If researcher doesn't know which treatment is received by each subject (double blind) +1
7.6. Do the researchers discuss the threats to validity?	Threats to validity are explained +1
8. Regarding *Conclusions*	**Total: 5 points**
8.1. Do the authors present results clearly?	If results are clear +1
8.2. Do the authors present conclusions clearly?	If conclusions are clear +1
8.3. Are the conclusions warranted by the results and are the connections between the results and conclusions presented clearly?	If conclusions are extracted from the results +1
8.4. Do the authors discuss their conclusions in relation to the original research questions?	If there is a link between RQ and conclusions +1
8.5. Do the authors discuss whether or how the findings can be transferred to other populations, or consider other ways in which the research can be used?	If there is a value for research or practice +1

In the *planning* step the protocol was defined, the details of which were explained in the previous section. In the *conducting* step we can see how the different documents were selected according to their relevance. The outcomes show the results after each step, such as the protocol review, or the number of papers that we had at a given time.

The protocol was developed by the three authors of this paper, and the searches were then carried out by the first author of the paper. The results of these were used by the second author to perform the first study selection, using abstracts and titles. The first author of the paper removed any duplicates and retrieved the data needed in the candidates for primary studies. After this first cycle, the protocol was improved by the three researchers. The study of the selected papers, along with their classification based on the full text, was carried out by the first author of the paper, who resolved any queries she had with the other two authors. A sample of 10 random papers was selected by the second and third authors of the paper to check the classification performed by the first author, and all three of us agreed on the classification of the sample.

Table 2.6. Systematic Mapping Study outline.

Chronology	Planning	Conducting	Reporting	Outcome
March 2010	Protocol development			Review protocol
April 2010		Data retrieval		Table with the metadata of the papers (808).
		Study selection on basis of abstracts and titles		Table with the metadata of the primary studies selected (148).
		Remove duplicates		Table with the metadata of the papers (85)
		Retrieval of the files of the primary studies		Repository of primary studies
May 2010	Protocol improvement	Pilot data extraction		Data extraction form with the classification scheme refined. 85 primary studies reviewed
		Study selection, quality assessment, and classification based on the full text		Data extraction form completed with the classification of papers (53).
		Resolution of queries in classification of primary studies in group		Revised classification of the primary studies (38).
		Data synthesis		
September 2010			Report on all the steps and activities carried out during the systematic mapping study process.	Final report

The *Planning* for the systematic mapping study began in March 2010, and papers published between 1997 and March 2010 were retrieved in April 2010. 808 papers were found (Figure 2.1). The title and abstract of each of the papers was examined and all those not dealing with empirical studies concerning the use of the UML in maintenance tasks were excluded, thus reducing the total to 148 papers. 63 duplicate papers (since there is some overlap between the electronic databases covered by the

different search engines, and some papers were therefore found by more than one of them) were discarded. The inclusion and exclusion criteria were then applied by reading the full text of each of the 85 remaining papers, leading to the discarding of 32 other papers.

We then detected that some of the empirical studies were included in more than one paper, so we also eliminated those 15 papers which contained results of empirical studies that had been summarized in other papers (we maintained the last published paper related to the same empirical study, which contained more details owing to the fact that they were journal papers). The final classifications were made on the final 38 papers, which reported66 primary studies (empirical studies), and these were then analyzed and the results interpreted.

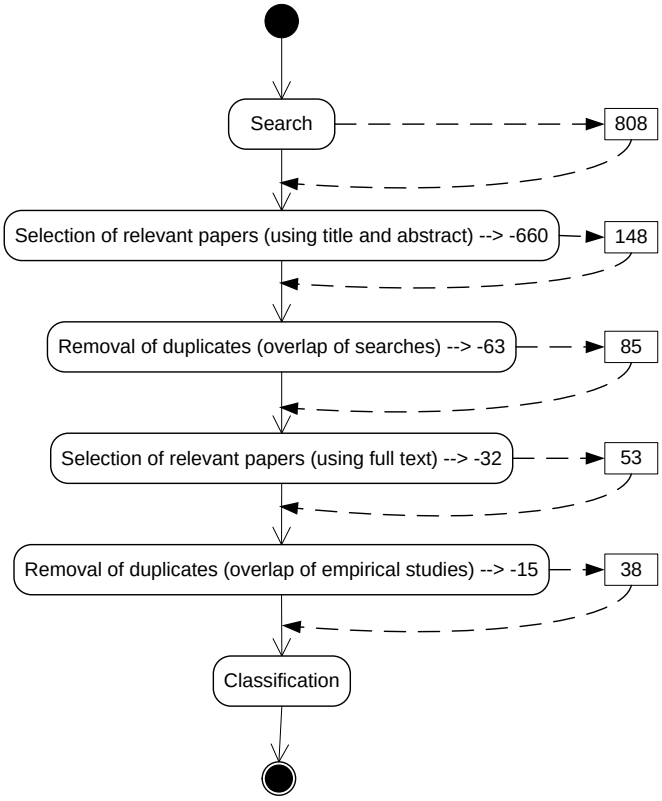

Figure 2.1. Selection process.

We were conscious that the search string was extremely long, and observed that, owing to the limitation of the search engines, such a long string could not be used directly. It was therefore necessary to tailor the search string to each digital library by splitting the original and then combining the results manually. Current search engines are not designed to support systematic literature reviews or systematic mapping studies. Unlike medical researchers, software engineering researchers need to perform

resource-dependent searches (Brereton et al., 2007). In order to alleviate, in part, some of the limitations of the search engines, we have used the tool known as SLR-Tool (Fernández-Sáez et al., 2010), which allowed us to refine the searches. More information on how the original search string was tailored to each digital library is shown in APPENDIX C. THE SEARCH STRINGS (related to Chapter 2)

2.5. REPORTING RESULTS AND DATA SYNTHESIS

Finally, the *reporting* step presents and interprets the results. In this section, we present the results of the systematic mapping study based on the 38 papers eventually selected. The structure of the results is based on the research questions which were set out above in Section 2.3. Data extracted from the papers reviewed were analyzed both quantitatively and qualitatively in order to answer the research questions.

2.5.1. Counting empirical studies

It is first necessary to explain some details about how the papers and the empirical studies were counted. Some papers collected in our investigation reported a single empirical study or a single replication. However, some other papers reported more than one experiment or replication in a single paper and others reported one or more replications together with an original study in a single paper. In these cases, for only one paper we counted each replication and original separately (i.e., we had more than one primary study), and this is why there are a different number of papers and empirical studies. In the remainder of this paper each paper will be called *Px*, in which *x* is the number of the paper. Each *Px* is a complete reference to a paper listed in APPENDIX A. LIST OF PRIMARY STUDIES (related to Chapter 2)

We started with 38 papers and obtained a total of 66 empirical studies, since some papers contained the results of more than one empirical study, as explained previously, and it is for this reason that we refer to "empirical studies" rather than "papers" when answering some of the research questions. Moreover, an empirical study may be related to more than one item from each of the categories defined, such as a paper related to the maintenance of class diagrams but also of sequence diagrams. In this case that primary study will therefore be counted twice. On the basis on this, the total number of empirical studies which appear in the result tables may therefore be greater than 66 (resulting in a "fictitious" total). The percentages are consequently calculated by using the "fictitious" total as the basis, rather than by taking the actual number of empirical studies (66) as a starting point. This is done to prevent percentages above 100%, which can sometimes hinder the understandability of the results.

2.5.2. Answers to the research questions

Firstly, we would like to remark that we found that only 2 papers ([P5] and [P8]) of the 38 are related to the use of UML diagrams in the maintenance of source code, and that the rest are related to the maintenance of the UML diagrams themselves. These two papers represent 3 of the 66 empirical studies mentioned above. Although they are the only papers related to the maintenance of source code and they are expected to

answer only the first part of our main research question, they also show results related to the maintenance of the UML diagrams themselves. As such, they also helped us to answer the second part of our main research question (i.e., in the following subsections they will be counted in the results for both parts of the question). In the following subsections we answer our research questions.

2.5.2.1. RQ1: Which diagrams are most frequently used in studies concerning the maintenance of UML diagrams or the maintenance of source code when using UML diagrams?

We analysed the studies in an attempt to find any reference to the 13 diagrams of the UML 2.3. The results which answer RQ1 are shown in Table 6, although those diagrams of which no evidence was found are omitted. 10.10% of the studies have been classified as not being related to a specific type of UML diagram. The type of diagram that is most frequently studied is the class diagram, in 34.34% of the studies. 17.17% refer to sequence diagrams, 16.16% to statechart diagrams, and 11.11% refer to collaboration diagrams. Only 8.08% of the studies selected relate to use case diagrams and 2.02% to activity diagrams, and only one study focused on deployment diagrams. No studies addressing any of the four new diagrams that were introduced with the UML 2.0 were found.

Table 2.7. Results per type of diagram.

Available diagram	Number of studies	Percentage	List of papers
Class diagrams	34	34.34%	[P3], [P5], [P8], [P9], [P12], [P13], [P14], [P18], [P19], [P20], [P22], [P24], [P26], [P28], [P29], [P32], [P35], [P36], [P37]
Sequence diagrams	17	17.17%	[P1], [P4], [P5], [P6], [P8], [P10], [P14], [P18], [P21], [P22], [P31], [P32], [P35], [P38]
Statechart diagrams	16	16.16%	[P1], [P4], [P7], [P21], [P22]
Collaboration diagrams	11	11.11%	[P1], [P4], [P14], [P21], [P22], [P29], [P34], [P35], [P38]
UML diagrams	10	10.10%	[P16], [P17], [P23], [P25], [P27], [P30]
Use case diagrams	8	8.08%	[P2], [P5], [P8], [P14], [P18], [P35]
Activity diagrams	2	2.02%	[P33]
Deployment	1	1.01%	[P35]
Total	**99**		

The low proportion of studies relating to use case diagrams is noticeable (Table 2.7). This may be explained by the fact that there are no studies addressing this type of diagram which are directly related to maintenance tasks. This low figure might also be

related to the origin of the diagrams. In some cases the diagrams are obtained from the source code, using reverse engineering, and in this case neither use case diagrams nor sequence diagrams are generally available when using open source tools. Furthermore, use cases say nothing about the structure of the system, and hence do not contain information that a maintainer needs to perform changes/modifications.

If we focus on the empirical studies that are related solely to the use of UML diagrams in the maintenance of code, all of these (3) used use case, class and sequence diagrams.

As mentioned above, the UML diagram that is studied most extensively is the class diagram, as seen in the results presented in (Dobing and Parsons, 2006), in which the most widely- used UML diagram in maintenance documentation is the class diagram. In (Dobing and Parsons, 2006) other rankings are provided based on different points of view or *key purpose* as is mentioned in that paper. We focus solely on the ranking related to maintenance documentation because it is the ranking which is most frequently related to that presented here. Moreover, our study places the sequence and statechart diagrams in high positions of use, which is consistent with the results provided in (Dobing and Parsons, 2006).

2.5.2.2. RQ2: Which dependent variables are investigated in the empirical studies?/ How are they measured?

The variables investigated when the mainainability of the UML diagams is studied are now shown, ordered by the type of diagram to which each is related: class diagrams (see Table 2.8), statechart diagrams (see Table 2.9), sequence diagrams (see Table 2.10), collaboration diagrams (see Table 2.11), use case diagrams (see Table 2.13), and activity diagrams (see Table 2.12). We also have another broader category: variables related to UML diagrams in general (see Table 2.14).

If these tables are observed it will be noted that the variable which is most widely-studied is the understandability of the class diagrams (22.64%), followed by the understandability of statechart diagrams (13.21%). Other common dependent variables are the modifiability (12.26%) and the analyzability (7.55%) of class diagrams, since these are considered to be sub-characteristics of maintainability. Maintainability is also studied as a whole (in class diagrams (0.94%) and in the whole system (2.83%)).In addition, there are several studies whose experiments address the understandability of other specific UML diagrams (11.32% of sequence diagrams and 9.43% of collaboration diagrams).

Table 2.8. Variables and measures for class diagrams.

CLASS DIAGRAMS				
Dependent variable	**Measure**	**Number of studies**	**Percentage**	**List of papers**
Understandability 24 studies 22.64%	time	16	15.09%	[P3], [P9], [P12], [P17], [P18], [P19], [P28], [P29], [P32], [P36], [P37]
	correctness	8	7.55%	[P9], [P18], [P19], [P20]
	F-measure	4	3.77%	[P26]
	accuracy	5	4.72%	[P17], [P24], [P29], [P36], [P37]
	effectiveness	6	5.66%	[P19], [P32]
	errors	2	1.89%	[P9], [P28]
	efficiency	2	1.89%	[P20], [P32]
	perceived comprehensibility	2	1.89%	[P3], [P12]
	relative time	1	0.94%	[P29]
Modifiability 13 studies 12.26%	time	11	10.38%	[P3], [P11], [P12], [P13], [P19]
	correctness	7	6.60%	[P13], [P19]
	effectiveness	7	6.60%	[P13], [P19]
	perceived comprehensibility	2	1.89%	[P3], [P12]
Analyzability 8 studies 7.55%	time	6	5.66%	[P3], [P11], [P12], [P13]
	perceived comprehensibility	2	1.89%	[P3], [P12]
	correctness	2	1.89%	[P13]
	effectiveness	2	1.89%	[P13]
Maintainability 1 study 0.94%	errors	1	0.94%	[P27]
Quality 1 study 0.94%	accuracy	1	0.94%	[P21]
Easy of construct 1 study 0.94%	accuracy	1	0.94%	[P21]

Table 2.9. Variables and measures for statechart diagrams.

STATECHART DIAGRAMS				
Dependent variable	**Measure**	**Number of studies**	**Percentage**	**List of papers**
Understandability 14 studies 13.21%	time	6	5.66%	[P4], [P7], [P21]
	efficiency	5	4.72%	[P7]
	effectiveness	5	4.72%	[P7]
	F-measure	6	5.66%	[P1], [P7]
	correctness	5	4.72%	[P7]
	accuracy	3	2.85%	[P4], [P21]

Table 2.10. Variables and measures for sequence diagrams.

SEQUENCE DIAGRAMS				
Dependent variable	**Measure**	**Number of studies**	**Percentage**	**List of papers**
Understandability 12 studies 11.32%	accuracy	8	7.55%	[P4], [P6], [P14], [P21], [P22], [P31]
	time	8	7.55%	[P4], [P14], [P18], [P21], [P22], [P38]
	correctness	2	1.89%	[P10], [P18]
	efficiency	1	0.94%	[P10]
	perceived comprehensibility	1	0.94%	[P14]
	F-measure	1	0.94%	[P1]
Quality of construction 1 study 0.94%	accuracy	1	0.94%	[P14]
	time	1	0.94%	[P14]
	perceived ease of construction	1	0.94%	[P14]

Table 2.11. Variables and measures for collaboration diagrams.

COLLABORATION DIAGRAMS				
Dependent variable	**Measure**	**Number of studies**	**Percentage**	**List of papers**
Quality of construction 1 study 0.94%	accuracy	1	0.94%	[P14]
	time	1	0.94%	[P14]
	Perceived ease of construction	1	0.94%	[P14]
Understandability 10 studies 9.43%	accuracy	8	7.55%	[P4], [P14], [P21], [P22], [P29], [P34]
	time	9	8.49%	[P4], [P14], [P21], [P22], [P29], [P34], [P38]
	relative time	1	0.94%	[P29]
	perceived comprehensibility	1	0.94%	[P14]
	F-measure	1	0.94%	[P1]

Table 2.12. Variables and measures for activity diagrams.

ACTIVITY DIAGRAMS				
Dependent variable	**Measure**	**Number of studies**	**Percentage**	**List of papers**
Complexity 2 studies 1.89%	errors	2	1.89%	[P33]
	number of elements	1	0.94%	[P33]

Table 2.13. Variables and measures for use case diagrams.

USE CASE DIAGRAMS				
Dependent variable	**Measure**	**Number of studies**	**Percentage**	**List of papers**
Understandability 2 studies 1.89%	accuracy	1	0.94%	[P2]
	time	1	0.94%	[P18]
	correctness	1	0.94%	[P18]
Retention 1 study 0.94%	accuracy	1	0.94%	[P2]
Problem-solving 1 study 0.94%	accuracy	1	0.94%	[P2]

Table 2.14. Variables and measures for the UML diagrams, in general.

DIAGRAMS (IN GENERAL)				
Dependent variable	**Measure**	**Number of studies**	**Percentage**	**List of papers**
Understandability 7 studies 6.60%	accuracy	3	2.85%	[P17], [P23]
	time	3	2.85%	[P17], [P23]
	errors	2	1.89%	[P16]
	efficiency	1	0.94%	[P30]
	F-measure	1	0.94%	[P25]
Error detection rate 2 studies 1.89%	errors	2	1.89%	[P16]
Learnability of modelled concepts 1 study 0.94%	errors	1	0.94%	[P15]

The variables for those empirical studies relatedto the maintenance of the source code when using UML diagrams (please recallthat these are also related to the maintenance of only the UML diagrams themselves) are presented in Table 2.15, and will be denominated as variables related tothe system in general (source code and

diagrams).Most of them are related to the time spent when performing maintenance tasks and the correctness of the solution (including its quality, measured through the number of errors).

Table 2.15. Variables and measures for the system (source code and diagrams).

SYSTEM				
Dependent variable	**Measure**	**Number of studies**	**Percentage**	**List of papers**
Maintainability 3 studies 2.85%	time	3	2.85%	[P5], [P8]
	correctness	3	2.85%	[P5], [P8]
	errors	3	2.85%	[P5], [P8]
Understandability 1 study 0.94%	F-measure	1	0.94%	[P35]

The variety of measures of dependent variables included in the 66 empirical studies presented in the 38 papers included in this systematic mapping study is relatively wide (Table 2.8-Table 2.15). On the one hand, there are several measures that have different names but measure the same concept (for example, in some papers the percentage of correct answers is called *correctness*, while other papers call this measure *effectiveness*). In order to construct Table 2.8-Table 2.15 and to count how many papers use the same measure, we have grouped those measures that look at the same concept under one name, so that the resulting number makes sense. The groups of measures with the same name, along with the definition of each measure and the papers in which they have been defined and used, are detailed in APPENDIX B.

As will be noted, most of the measures are based on objective measures, such as the number of correct answers, the number of questions, or the time spent on the tasks, in addition to different calculations based on all of these. On the other hand, a minority of studies use subjective variables, related to the subjects' perceptions of the variable measured.

We were surprised to discover that none of the studies considered investigated the use of the UML in productivity in software maintenance, since productivity is often a crucial factor which all software development organizations attempt to maximize. One of the reasons for this might be that measuring the impact of using the UML on productivity in a project is no trivial task: it can be both expensive and difficult, although there were two studies, reported in [P5] and [P8], which did investigate the impact of the UML on software maintenance in an experimental setting. We would like to stress that productivity is an indirect measure which needs some recognition and requires some form of model for its derivation, perhaps based on other direct measures but also by using some environmentally dependent factors. Productivity can be derived from various final measures, which could make a comparison difficult. Another explanation for the lack of studies on the use of the UML in productivity in software maintenance might be that UML diagrams are rarely consulted in maintenance tasks. A poor diagram/code correspondence could explain why the UML

diagrams for maintenance are ignored. But it is also very likely that UML diagrams are consulted or not irrespective of their high or low correspondence to the code, as is explained in (Nugroho and Chaudron, 2008).

For each study it is important to know what kind of tasks the subjects had to perform in order to understand why one dependent variable is used rather than another. Most of the studies that were found perform tasks to test the comprehension of the diagrams. This means that most of the tasks which are performed by the subjects involve answering questionnaires. There are also some studies in which the tasks to be carried out are those of modifying a diagram so that it meets certain requirements.

Table 2.16. Results per duration.

Duration (in minutes)	Number of studies	Percentage	List of papers
0-60	6	9.09%	[P1], [P2], [P15], [P19], [P30], [P38]
61-120	12	18.18%	[P10], [P13], [P14], [P18], [P20], [P23], [P29], [P33]
121-300	8	12.12%	[P16], [P24], [P26], [P28], [P31]
301-1000	2	3.03%	[P5]
"+" 1000	8	12.12%	[P3], [P8], [P11], [P12], [P13]
Not specified	26	39.36%	[P4], [P6], [P7], [P9], [P16], [P17], [P19], [P21], [P22], [P25], [P27], [P32], [P35]
Others	4	6.06%	[P4], [P33], [P34], [P36], [P37], [P38]
Total	**66**		

It is important to note that in many studies (almost 40%) the duration of the tasks in not specified, and if it is specified, the duration of the task is usually short, from 1 to 2 hours in length (Table 2.16) to avoid the situation of subjects becoming tired and fatigued; this fatigue would be a threat to the internal validity of the studies. In contrast, there are some uncontrolled experiments in which the presence of the supervisor is not necessary and the subjects have one week to complete the tasks (we consider these to be experiments of 168 hours, i.e., 7 days multiplied by 24 hours per day). There are studies which indicate the time in a measure that cannot be translated into minutes (for example, papers that measure the time taken to do the experiments in "sessions", in which we do not know the length of each one). These studies have not been taken into account in our calculations (as unconstrained time studies) and they are included in the "others" category. This signifies that, of all the studies found, the average time taken amounts to 2166.53 minutes (36h, approximately).

2.5.2.3. RQ3: What is the state-of-the-art in empirical studies concerning the maintenance of UML diagrams or the maintenance of source code when using UML diagrams?

This subsection presents several issues related to the state-of-the-art in empirical studies concerning the maintenance of the UML diagrams themselves, or to the maintenance of source code when using UML diagrams. These are the following: the type of empirical study (i.e., the empirical methods), the kind of context in which the

empirical studies were executed, the kind of participants in the empirical studies (i.e., the subjects), what was maintained during the study (i.e., the object maintained), the type of systems used during the studies, the treatments of the studies (i.e., the independent variables), and finally, the quality of the empirical studies and papers.

There are many research methods to choose from when carrying out any investigation. We focused only on those studies that are carried out empirically, as dictated by one of the inclusion criteria. The results of the validation classification method are shown in Table 2.17. We would remind the reader that the number of empirical studies is higher than 38 owing to the fact that several papers fall into more than one category, and we therefore have 66 empirical studies (for example, one paper contains both an experiment and a case study). 95.45% of the studies report the results of a controlled experiment, as is shown in Table 2.17 (note that all of the empirical studies concerning the use of the UML in the maintenance of source code are in this category). This finding shows the need to conduct more case studies, as this is a kind of experimentation that deals with real environments and real projects.

Table 2.17. Results per empirical method.

Empirical method	Number of studies	Percentage	List of papers
Experiment	63	95.45%	[P1], [P2], [P3], [P4], [P5], [P6], [P7], [P8], [P9], [P10], [P11], [P12], [P13], [P14], [P15], [P26], [P18], [P19], [P20], [P21], [P22], [P23], [P24], [P25], [P26], [P28], [P29], [P30], [P31], [P32], [P33], [P34], [P36], [P37], [P38]
Case study	3	4.55%	[P27], [P27], [P35]
Survey	0	0.00%	-
Action Research	0	0	-
Total	**66**		

According to (Yin, 2002), a "case study is an empirical inquiry that investigates a contemporary phenomenon within its real-life context, especially when the boundaries between the phenomenon and context are not clearly evident". Bearing this definition in mind, although some studies claimed that a case study was being presented, they were removed, because in actual fact they contained only an example.

The context in which the studies were carried out could be an industrial context or a laboratory (Table 2.18). Most of the studies found regarding the maintainability of UML diagrams (83.33%) are the results of experiments that have been conducted in laboratories within academic environments. In the case of the studies on the maintenance of source code, all of these were performed within a laboratory environment. There are also some papers that present the results of empirical studies in industrial settings, but the percentage of this type of studies is very low (4.55%). Those studies that indicated that the subject under study could do the test at home have been considered as having been carried out within a non-controlled context (12.12%).

Table 2.18. Results per context.

Context	Number of studies	Percentage	List of papers
Laboratory	55	83.33%	[P1], [P2], [P4], [P5], [P6], [P7], [P8], [P9], [P10], [P13], [P14], [P16], [P18], [P19], [P20], [P21], [P22], [P23], [P24], [P25], [P26], [P28], [P29], [P30], [P31], [P32], [P33], [P34], [P36], [P37], [P38]
Non-controlled	8	12.12%	[P3], [P11], [P12], [P13], [P15]
Industrial	3	4.55%	[P17], [P27], [P35]
Total	**65**		

Table 2.19. Results per type of subject.

Type of subjects	Number of studies	Percentage	List of papers
Students	59	77.63%	[P1], [P2], [P3], [P4], [P5], [P6], [P7], [P9], [P10], [P11], [P12], [P13], [P14], [P15], [P16], [P17], [P18], [P19], [P20], [P21], [P22], [P23], [P24], [P25], [P26], [P28], [P29], [P30], [P31], [P32], [P33], [P34], [P36], [P37], [P38]
Practitioners	9	11.84%	[P6], [P7], [P8], [P16], [P17], [P27], [P29], [P35], [P38]
University Lecturers	8	10.53%	[P3], [P7], [P11], [P12], [P26], [P30]
Total	**76**		

The average number of subjects used in the empirical studies in the papers found is 41.19. Table 2.19 shows what types of subjects were used in the empirical studies. The majority of empirical studies (77.63%) tended to be carried out with undergraduate students, in the third, fourth or fifth year of the Computer Science degree. This is not necessarily inappropriate (Budgen et al., 2011a; Kuzniarz et al., 2013), because the UML is intended to support design tasks, and students' design skills are likely to be similar to those of non-expert professionals. A considerably lower percentage of empirical studies was carried out by members of the university teaching staff (10.53%) or by practitioners (11.84%).

These results show that there is a need to perform more empirical studies with practitioners in order to confirm whether the results obtained with students are also valid with the former type of subjects.

We shall now go on to discuss the type of object(s) that had to be maintained. Software maintenance tasks have always required some changes to be made to the source code (Table 2.20). There is the possibility of using diagrams to maintain the code and of updating these diagrams to reflect the changes (16.67%), or there is the option in which the diagrams are the only elements maintained (83.33%). This second option makes sense when attempting to obtain empirical studies about the

understandability of the diagrams. There are no studies that deal with the maintenance of the code of a system supported by the use of the UML diagrams, but we did find studies in which the UML diagrams are not maintained, which is what appears to occur most often in practice.

Table 2.20. Results per object to maintain.

Object to maintain	Number of studies	Percentage	List of papers
Diagrams	55	83.33%	[P1], [P2], [P3], [P4], [P6], [P7], [P9], [P10], [P11], [P12], [P13], [P14], [P15], [P16], [P17], [P18], [P19], [P20], [P21], [P22], [P23], [P24], [P25], [P28], [P29], [P30], [P32], [P33], [P36], [P37], [P38]
Code + Diagrams	11	16.67%	[P5], [P6], [P26], [P27], [P31], [P34], [P35]
Code	0	0.00%	-
Total	**66**		

Owing to the low percentage of empirical studies that examine the maintenance of both diagram and code, there is no evidence to allow us to really know whether the results obtained in studies with isolated diagrams or isolated code can be generalized to real environments. There is thus a need to carry out more studies of this type, which deal with the maintenance of only the UML diagrams themselves as part of the maintenance of the entire system. The performances of maintainers when using up to date diagrams or older versions of the documentation, e.g., that originate from the design of a system, also need to be compared. In our opinion, the degree of correspondence between diagrams and code could influence some of the maintenance tasks. It is supposed that better results would be obtained when this correspondence is high.

Table 2.21. Results per type of system.

Type of system	Number of studies	Percentage	List of papers
Synthetic	51	73.91%	[P1], [P2], [P3], [P4], [P5], [P6], [P7], [P9], [P10], [P11], [P12], [P13], [P14], [P15], [P16], [P17], [P18], [P19], [P20], [P21], [P22], [P23], [P24], [P29], [P31], [P32], [P33], [P34], [P38]
Real	18	26.06%	[P8], [P17], [P19], [P25], [P26], [P27], [P28], [P30], [P35], [P36], [P37]
Total	**69**		

At this point, we should discuss the origin of the materials used in the studies. Most of the empirical studies that were found (73.91%) used diagrams made from synthetic systems such as prototypes or systems developed specifically for the study (Table 2.21). Only 26.06% of the diagrams used represent real systems in operation. There is

a need to perform more empirical studies with real systems, since most studies address diagrams of small systems, using convenience systems such as a library, ATM, etc., which may not accurately represent the behaviour of large industrial systems.

Table 2.22. Treatments in the empirical studies.

Treatments	Description	Number of studies	Percentage	Papers
UML vs. UML	composite states vs. non-composite states (or different nesting levels)	10	15%	[P7]
	diagrams with stereotypes vs. diagrams without stereotypes	9	14%	[P10], [P25], [P26], [P29], [P31], [P37]
	sequence diagrams vs. collaboration diagrams vs. statecharts	3	5%	[P4], [P21]
	sequence diagrams vs. collaboration diagrams	3	5%	[P14], [P38]
	high Level of Detail vs. Low Level of Detail	1	2%	[P20]
	diagrams with geons vs. diagrams without geons	1	2%	[P15]
	animated diagrams vs. non-animated diagrams	1	2%	[P6]
Measure *X* vs. measure *Y*	Values of different measures calculated using the diagrams	14	21%	[P3], [P11], [P12], [P13], [P19], [P28]
UML vs. other modeling languages	OML vs. UML	2	3%	[P22]
	EPC vs. UML	2	3%	[P33]
	OPM vs. UML	1	2%	[P24]
	UML vs. UML-B	1	2%	[P23]
	UML-B vs. UML + event-B diagrams	1	2%	[P23]
UML vs. non UML	Presence of UML diagrams vs. absence of UML diagrams	5	8%	[P1], [P5], [P8], [P30]
Using or not a tool	using a tool (metricViewEvolution) vs. Not using a tool	3	5%	[P17], [P18]
Layout *X* vs. layout *Y*	different layouts	2	3%	[P9], [P36]
Defect *X* vs. defect *Y*	Presence of different kinds of defects in the UML diagrams	2	3%	[P16]
Notation *X* vs. notation *Y*	different notations (on the same diagram)	1	2%	[P34]
Diagrams vs. text	use case diagrams vs. text cases	1	2%	[P2]

Treatments	Description	Number of studies	Percentage	Papers
Forward diagrams vs. RE diagrams	Forward-designed diagrams vs. Reverse-Engineered diagrams	1	2%	[P27]
Retrieval method *X* vs. retrieval method *Y*	Different retrieval methods of the UML diagrams	1	2%	[P35]
Transformation rule *X* vs. transformation rule *Y*	Different transformation rules between the UML diagrams	1	2%	[P32]
	Total	**66**		

We also extracted some information about the independent variables that were used in the empirical studies, i.e., whose treatments were being used in the various experimental studies. Table 2.22 shows that most of the empirical studies (42%) compare different aspects of the UML diagrams, such as diagrams with stereotypes vs. those without them, different levels of details in the diagrams, etc. This is followed by the values of different metrics which measure some aspects of the UML diagrams, such as complexity, size, etc. (21%), the comparison between the UML and other modelling languages (11%), the presence or absence of the UML diagrams (8%), and so on. All the papers related to the use of the UML diagrams when maintaining the source code are in this last category.

Finally, we present the quality assessment results obtained by applying the quality criteria shown in Table 2.5 to the primary studies. These papers were evaluated to discover whether or not they covered these criteria, and the papers were therefore scored by applying the quality measures shown in Table 2.5. It was possible for a primary study to obtain a maximum of 40 points. Based on that number, we decided to consider three categories: high quality (from 25 to 40 points, i.e. papers with more than 60% of the total points), medium quality (from 16 to 24 points, i.e. papers with 40- 60% of the total points), and low quality (from 0 to 15 points, i.e. papers with less than 40% of the total points).

Most of the 38 papers containing primary studies obtained a relatively high score in this quality assessment, as is shown in Table 2.23 (note that all of the papers related to the maintenance of source code when using the UML are in this category).

If we focus on the first quality criterion, i.e., that related to the description of *Aims and Objectives*, we can state that most of the papers obtained a high score because only 1 paper obtained less than 2 points out of 5. The majority of the papers had a good description of the context in which the studies was performed, since 73.68% obtained the maximum score (5) in the criterion concerning the description of the *Context*. This might be owing to the fact that the results of experiments in Software

Engineering cannot be generalized to the whole community and the results are only valid for specific contexts, so they should be commented on in a detailed manner. If we focus on the criterion concerning the description of the *Design of the Experiment*, almost half of the papers (39.47%) provided a complete description of the design of the paper. The same percentage of papers forgot to justify the research design or the description of the treatments, more or less in the same proportion. It is quite surprising that 21.08% of the papers did not obtain any score in this criterion because they did not describe the design of the experiment. Upon focusing on the next criterion, *Control Group*, we can state that half of the papers (52.63%) clearly describe the control group used to compare the treatments and the other half do not. If we focus on the *Data Collection* criterion, we can consider that the majority of papers obtained high scores because only 18.42% obtained less than 3 points out of 5, but almost none of them described whether they used a quality control method to ensure the consistency, completeness and accuracy of the data collected. The next criterion concerns the description of the *Data Analysis Procedures*, in which most of the papers obtained a medium score (50% of the papers obtained from 4 to 6 point out of a maximum of 10). In this case we cannot state that these papers are particularly good or bad at describing this, but it is important to highlight that none of the primary studies provide references to the raw data used to test the results. One important section in papers is that concerning *Threats to Validity/Bias*, which is related to our next criterion. The maximum score obtained by papers in this category is 4 out of 7 points. As part of its analysis, we consider it important to note that almost no primary study studied the influence between researchers and participants, or the implications of developing a special system to work on during the research, both of which might influence the validity of the results. We would also like to underline the need to clarify whether or not the review was double blind, i.e., whether the researchers know which treatment is received by each subject, in order not to influence the results when checking their responses. Our last criterion is related to the description of the *Conclusions* of the study, in which the majority of papers obtained a high score (63.16% obtained 4 out of 5 points).

Table 2.23. Quality of primary studies.

Quality	Number of papers	Percentage	List of papers
Low	4	10.53 %	[P6], [P27], [P32], [P35]
Medium	14	36.84%	[P1], [P3], [P4], [P7], [P12], [P15], [P17], [P19], [P23], [P28], [P30],[P31], [P34], [P37], [P38]
High	20	52.63%	[P2], [P5], [P8], [P9], [P10], [P11], [P13], [P14], [P16], [P18], [P20],[P21], [P22], [P24], [P25], [P26], [P29], [P33], [P36]
Total	**38**		

2.5.2.4. RQ4: Which of the factors studied influence the maintainability of a system (source code and diagrams)?

We have extracted the different factors that can influence the maintainability of systems from the studies analyzed in this systematic mapping study (Figure 2.2). The factors are shown in rounded boxes, and the rectangular boxes contain categories that we have added in order to classify all the factors. A factor that has a positive influence is represented with the symbol *plus* (+), and the negative influence is indicated with the symbol *minus* (-).A number related to a further explanation in the following paragraph is shown in brackets. As mentioned previously, it is well known that understandability directly influences maintainability (Briand et al., 2001a; Deligiannis et al., 2002; Genero et al., 2007; Harrison et al., 2000). We therefore assume here that those factors that are related to understandability are also related to maintainability.

The content of Figure 2.2 is explained thus:

- The maintainability of a system is influenced by the maintainability of its source code and its documentation, which can consist of a text or models – UML or non UML models. OPM (non UML) might be considered to be a better notation but only in the context of modelling the dynamic aspect of Web applications. Other notations that are extensions of UML, like UML-B, have a positive influence on maintainability since they facilitate understanding.
- A maintainer's skill also affects the maintainability of a system, signifying that when a maintainer has some experience, this will have a positive influence on the maintenance of the system.
- The maintainability of the source code is negatively influenced when the complexity of the system is high, but it is positively influenced by a correct traceability from the diagrams to the source code.
- The maintainability of the UML diagrams is positively influenced by the presence of Reverse Engineered diagrams. These, combined with forward design diagrams, help to detect possible errors in the system, which facilitates its maintenance.
- The maintainability of the UML diagrams is positively influenced by the availability of use case, class, sequence, activity and statechart diagrams, and also whether the diagrams contain stereotypes that detail certain characteristics of their elements. All these characteristics are related to the way in which the model is represented.
- Also related to the representation of models is the fact that the level of detail in the UML diagrams additionally affects the maintainability of systems (source code and diagrams), making it ideal to have a higher level of detail.
- The use of composite states also improves the understandability of the UML statechart diagrams. However, a high nesting level of composite state in the UML statechart diagrams negatively influences the understandability of these diagrams.
- With regard to the way in which models are visualized, the availability of interactive views or animations to improve the diagrams improves the visualization of a UML model, thus improving its maintainability. What is more a proper distribution of the elements (an aesthetic diagram layout) improves the

maintainability of a UML diagram. However, the use of textual use cases has a negative influence on maintainability.

Details of how the classification process of the factors was carried out are set out below. The maintainability of a whole can be considered as the sum of the maintainabilities of each of its parts. In this case, only code and diagrams (as part of the documentation of the system) were considered as part of a system.

The maintainability of the diagrams consists of some characteristics that are directly related to the diagram and others that the reader of the diagram introduces. A diagram can be influenced by its representation (the diagram itself, what is represented) and the way in which it is presented to the reader, i.e., its visualization, or its origin.

The maintainability of the code is influenced by both its own characteristics and the characteristics from the diagrams (since these provide complementary information about the code). In addition, the reader of the code introduces some influential factors.

In the following lines we will explain which paper refers to each factor, referencing the numbers that appear in Figure 2.2:

(1) Positive influence of some diagrams:
   a. Class diagrams: [P5], [P8] and [P22]
   b. Statechart diagrams: [P21] and [P22]
   c. Sequence diagrams: [P1], [P4], [P5], [P8], [P14], [P21], [P22] and [P38]
   d. Activity diagrams: [P33]
   e. Use case diagrams: [P2], [P5] and [P8]

(2) Positive influence of stereotypes: [P10], [P24], [P25], [P29] and [P31].

(3) Negative influence of aggregations as a kind of relationship in class diagrams: [P32].

(4) Influence of composite states: the use of composite states is a positive influence, but if the nesting level is high, the influence is negative [P7].

(5) Positive influence of a high level of detail: [P20].

(6) Positive influence of aesthetic quality or layout: [P9], [P30], [P34], [P36] and [P37].

(7) Positive influence of interactive views or the use of animations: [P6], [P15], [P17] and [P18].

(8) Negative influence of the use of textual use cases: [P2].

(9) Negative influence of the defects in the diagrams: [P16].

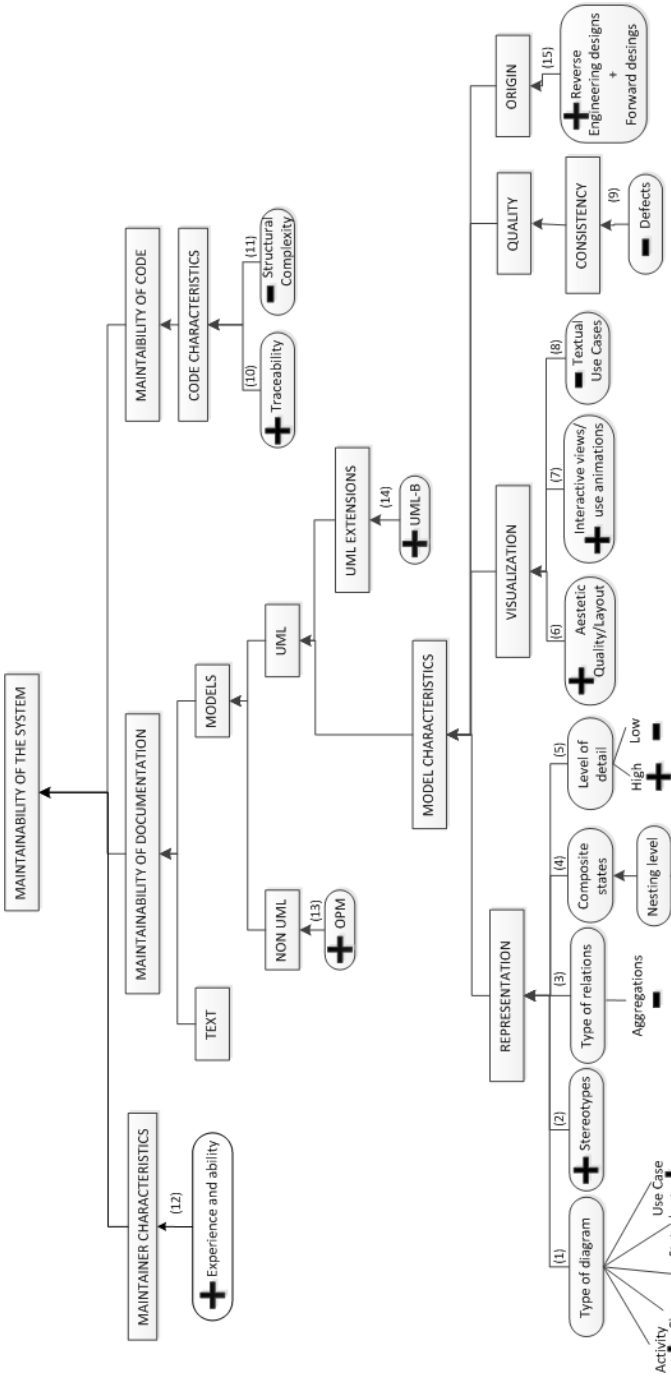

Figure 2.2. Factors that influence the maintainability of a system.

(10) Positive influence of traceability from diagrams to code: [P35].

(11) Negative influence of the structural complexity: [P3], [P7], [P11], [P12], [P13], [P19], and [P28].

(12) Positive influence of maintainers' experience and ability: [P26]

(13) Positive influence of other notations compared to UML in modelling the dynamic aspect of Web applications: [P23].

(14) Positive influence of UML extensions (UML-B): [P27].

(15) Positive influence of the presence of diagrams extracted from Reverse Engineering: [P24]

2.5.3. Additional results

The results obtained from the classification of papers are presented here within the "others" category.

Figure 2.3 shows that every year an almost constant number of new publications related to the topic of this study appear. This figure may show that interest in this subject has been growing over time, reaching its highest points in 2009. We should point out that the number of papers in 2010 is small, because the search was only performed until March 2010.Results reveal that there is a mean of almost 5 papers published on this topic per year.

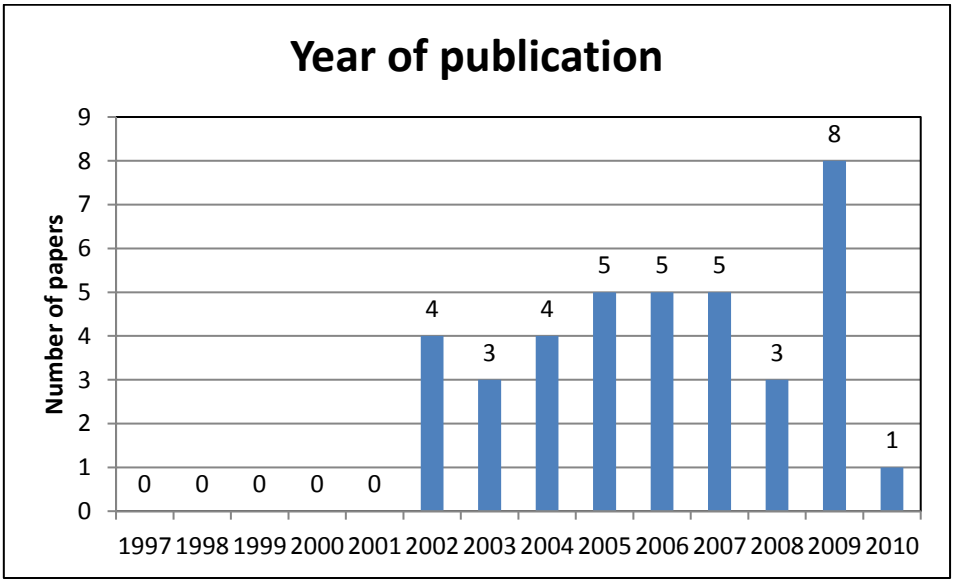

Figure 2.3. Number of papers per year.

When analyzing the types of publication, we found that 39.47% of the papers (15 papers) were published in conferences, 47.37% in journals (18 papers) and 13.16% in workshops (5 papers). The first paper in a journal appeared in 2002, with this figure increasing over the following years to a maximum in 2009, when it reached its highest

level of 5 papers. This coincides with one of the years with the highest number of publications (Figure 2.3). The use of UML diagrams in maintenance tasks has been judged to be a "hot topic", given the number of publications. The field is nonetheless quite mature, as is demonstrated by the percentage of journal papers.

Table 2.24. Number of papers per type of publication.

Journal/Conference	Number of papers	Percentage
Information and Software Technology	4	10.53%
IEEE Transactions on Software Engineering	3	7.89%
Empirical Software Engineering	2	5.26%
International Conference on Program Comprehension	2	5.26%
International Symposium on Empirical Software Engineering	2	5.26%
International Symposium on Empirical Software Engineering and Measurement	2	5.26%
International Workshop on Visualizing Software for Understanding and Analysis	2	5.26%

Table 2.24 shows only the publications with the largest number of papers related to the topic being studied. The first three positions are occupied by journals: Information and Software Technology (4 papers), IEEE Transactions on Software Engineering (3 papers), and Empirical Software Engineering (2 papers) which together represent almost 25% of the total. The conferences with the highest number of papers are the International Conference on Program Comprehension, the International Symposium on Empirical Software Engineering and International Symposium on Empirical Software Engineering and Measurement, all of them with 2 papers, and each one of them representing nearly 6% of the total. The workshop with the highest number of papers is the International Workshop on Visualizing Software for Understanding and Analysis, with 2 papers.

Table 2.25. Diagrams obtained from RE.

Diagrams from RE	Number of studies	Percentage	List of papers
Yes	10	15.15%	[P25], [P26], [P27], [P28], [P30], [P36], [P37]
No	56	84.85%	[P1], [P2], [P3], [P4], [P5], [P6], [P7], [P8], [P9], [P10], [P11], [P12], [P13], [P14], [P15], [P16], [P17], [P18], [P19], [P20], [P21], [P22], [P23], [P24], [P29], [P31], [P32], [P33], [P34], [P35], [P48]
Total	**66**		

It is also important to note that only 15.15% of the systems used in the empirical studies are obtained from a reverse engineering (RE) process, while 84.85% of the diagrams used are created during the development process (Table 2.25).

2.6. DISCUSSION

This systematic mapping study has discovered 38 relevant papers (containing 66 empirical studies) in peer-reviewed journals, conferences, and workshops, and has classified them in order to obtain responses to the research questions presented, which are briefly summarized below:

- *RQ1 asked:* Which diagrams are most frequently used in studies concerning the maintenance of UML diagrams or the maintenance of source code when using UML diagrams? The results show a clear order, which indicates the relative importance that researchers attach to 3 diagram types when they study the maintenance of the UML diagrams themselves or also how they study the use of UML diagrams when performing maintenance on the source code: class diagrams (34%), sequence diagrams (17%) and statechart diagrams (16%). Some studies performed partial comparisons of the understandability of one type of diagram versus another. The three aforementioned diagrams are reported to contribute most to understandability. The low occurrence of studies relating to use case diagrams (8%) could be explained by the fact that there are no studies addressing this type of diagrams which are directly related to maintenance tasks (they are always presented with other UML diagrams) . This low rate could also be related to the origin of the diagrams. In some cases (about 15%), the diagrams are obtained from the code by using reverse engineering. In this case, the use case diagrams are not generally available. Furthermore, owing to their high level of abstraction, use cases say nothing about the structure of the system, and hence do not contain information that a maintainer needs to perform changes/modifications which tend to be a lower level of information as regards detail.
- *RQ2 asked: Which variables are investigated in the empirical studies?/ How are they measured?* Most of the empirical studies found which focused on the maintenance of the UML diagrams themselves concentrated on measuring the understandability of class diagrams (23%) or statechart diagrams (13%). The measures used for this dependent variable are related to the time spent by the subjects in understanding the UML diagrams and the subjects' effectiveness when performing the understandability tasks. There are some more isolated studies focusing on the use of the UML diagrams in maintenance tasks. In these cases, the measures used are, apart from time, the correctness of the solutions proposed and the quality of the code. It is supposed that a better understanding of the diagram correlates with a better understanding of the system, and that this should positively influence the maintenance of the source code. However, sufficient work with which to validate this assumption is not available. More studies are needed which deal with the influence of UML diagrams on the maintainability of source code.
- *RQ3 asked:* What is the state-of-the-art in empirical studies concerning the maintenance of UML diagrams or the maintenance of source code when using UML diagrams? 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 three perspectives:

- ***How?:***
  - ***How*** *is the maintenance of the UML diagrams studied?*

    Most of the studies that were found present results of controlled experiments (95%). This is a well-known way in which to validate data, but the field would benefit (in terms of generalizability) from the additional performance of case studies. Industrial data or real projects should be analyzed to confirm the results obtained in the laboratory context.
- ***Where?:***
  - ***Where*** *are the empirical studies carried out?*

    We now know that most of the studies performed are controlled experiments. These studies can be considered as only academic results, since they were carried out in a laboratory context (83%), so it is also necessary to perform more empirical studies in industrial contexts to corroborate the academic results.
- ***What?:***
  - ***What*** *types of subjects have been used in empirical studies?*

    The subjects that performed the tests are mostly students (78%). A minority of studies involved members of the university teaching staff (11%) or practitioners (12%). This fact reveals that more empirical studies with practitioners are necessary to strengthen the external validity of the results. It would thus be feasible to ascertain whether or not the findings obtained with students also hold for practitioners.
  - ***What*** *is maintained in the empirical studies?*

    If we focus on the results obtained in this systematic mapping study, we can see that most of the studies are related to the maintenance of only the UML diagrams themselves (83%), rather than to the UML diagrams and the code (14%). It is also important to highlight that most of the diagrams used represent prototypes of systems or very simple systems (74%). Using diagrams from true complex systems when performing maintenance tasks would help to test whether the UML has specific benefits. It is also important to note that those experiments in which tasks are related to the maintenance of the code, rather than simply maintaining a diagram, are more representative of the current situation in industry. There should be more studies which deal with the maintenance of the UML diagrams themselves as part of maintaining an entire system. There also needs to be a comparison of the maintainers' performance when using up to date diagrams as opposed to using older versions of the documentation - e.g., that originate from the design of a system. In our opinion, the degree of correspondence between diagrams and code could have an influence

on some maintenance tasks. It would be logical to expect to obtain better results when this correspondence is high.

- ***What** are the treatments in the empirical studies?*

  Most of the empirical studies (42%) attempt to compare different aspects of the UML diagrams, for example diagrams with stereotypes vs. without them, different levels of details in the diagrams, etc.

- ***What** is the quality of the papers found?*
- More than half of the papers (53%)obtained a relatively high score in this quality assessment (note that all of the papers related to the maintenance of source code when using the UML are in this category), and only 9% obtained a low quality score. The fact that most of the papers obtain the maximum points related to the description of the aims of the research and its context is worth noting. None of the primary studies, in contrast, provide a reference to the raw data used to test the results.

- *RQ4 asked: Which of the factors studied influence the maintainability of a system (source code and diagrams)?* These results are summarized in a classification tree of factors in Figure 2.2. It will be observed that the presence of some specific diagrams, such as the use of stereotypes and a good, correct layout, positively influences the maintainability of a system. There are also other factors, such as a high structural complexity of the system, a low level of detail in diagrams, a high nesting level of composite states or the presence of defects in diagrams, which have a negative influence on the maintainability of a system.

It is also noteworthy that only two papers are specifically related to empirical studies concerning the use of the UML in maintenance (modification) tasks:

- The first of these is [P5], which presents the results of two controlled experiments carried out with students from different universities. [P5] reports that the time taken to make changes in the source code is less when the UML diagrams are used than when they are not used, while if the time taken to perform the corresponding modifications to diagrams is included, there is no significant difference. In both cases, however, the quality of the modifications is greater when the subjects have UML diagrams.
- [P8] presents the results of a controlled experiment carried out with professionals. In both this paper and that mentioned above, the time taken to perform the modifications to the system, the time spent on maintaining the diagrams and the quality of the proposed modifications are measured. This study [P8] does not find any significant difference in the time spent on performing changes, but the authors do find that the quality of the changes is higher for the group of subjects with UML diagrams, as is the case in [P5].

2.7. THREATS TO VALIDITY

We have classified the threats to validity on this study by following the classification provided by Wohlin et al. (Wohlin et al., 2012). The main threats to the validity of a systematic mapping study are publication selection bias (construct validity), inaccuracy in data extraction (construct validity), and misclassification (conclusion validity) (Sjøberg et al., 2005).

With regard to the *construct validity*, we considered six digital sources, which included journals, conferences and workshops which are relevant to software engineering. The scope of journals and conferences covered in this systematic mapping study is sufficiently wide to attain a reasonable completeness in the field studied. We did not include additional papers such as grey literature (technical reports, PhD thesis, etc.), and limited ourselves to peer-review publications. We believe that we have achieved a reasonably complete coverage, as most grey literature either has its origins in peer-reviewed papers or appears in what will eventually become peer-reviewed papers; it may, however, be the case that both of these circumstances are true for a given piece of grey literature. Some relevant papers might exist which have not been included (which it might be possible to extract with the use of a snowballing process), although our knowledge of this subject is such that we do not believe that there are many of these. We performed an automated search on 6 digital libraries in order not to rule out papers from conferences or journals which deal with topics of interest but may not be well-known sources. This could be a threat to the validity of this work because manual searchers seem to be more helpful than those which are automated but this requires a previous knowledge of the source used in the search, as is presented in the results of (Kitchenham et al., 2010), but this work was published after we had performed the search, and we have not therefore been able to take these results into consideration. To help ensure an unbiased selection process, we defined research questions in advance, organized the selection of papers as a multistage activity, involved three researchers in this activity and documented the reasons for inclusion/exclusion, as suggested in (Liu et al., 2005). As was discussed above, the decisions to select the papers to be included as primary studies in this systematic mapping study were made by multiple researchers, and rigorous rules were followed. A further challenge was that there is no keyword standard that we are aware of which distinguishes between different quality characteristics, nor are there methods in empirical software engineering that could be used to extract quality characteristics and research methods in a consistent manner.

Moreover, the duplication of papers is a potential threat to frequency counts and to the statistics in this systematic mapping study. The structure of the database managed by the SLR-Tool (Fernández-Sáez et al., 2010), which was used to perform this systematic mapping study, is designed to handle duplication, but one threat would be that of duplication going undetected. However, at least two individuals have read through all the relevant papers without detecting further duplicates. We also found it quite difficult to manage the duplication of empirical studies performed by the same author but which are reported as a part of other studies, i.e., different papers had a part

of their contents in common. We examined them exhaustively in order to attempt to detect whether or not they were the same study, following a fixed procedure, but the elimination or otherwise of possible duplications might be a threat.

The fact that we also considered the term understandability as an alternative term for maintainability, which a priori is not a real synonym or sub-characteristic of it based on the ISO 25000 (ISO/IEC, 2008), might be a threat of the validity of our work. However, we based our decision on the results of previous works which judge understandability to be a factor that influences maintainability (Briand et al., 2001a; Deligiannis et al., 2002; Genero et al., 2007; Harrison et al., 2000).

With regard to *conclusion validity*, we would like to comment that when extracting data from papers there is a certain degree of subjectivity in terms of what is and what is not determined to be related. Furthermore, bias can affect the interpretation of the results. The data was extracted from the papers by one researcher and checked by another. When necessary, disagreements were resolved through discussion by involving the third author. Data extraction and classification from prose is difficult at any time, and the lack of standard terminology and standards could very well result in a misclassification. We believe, however, that the extraction and selection activity was rigorous and that it followed the guidelines provided in (Brereton et al., 2007). The use of multiple experts to perform the classification also reduced the risk of misclassification.

2.8. CONCLUSIONS

More than fifteen years on from when the UML was first introduced in 1997, it would be useful for the software industry to gather empirical evidence of use of the UML in the software development life cycle, specifically in software maintenance, which is the most resource-consuming phase. With that need to gather such information in mind, this paper presents a systematic mapping study on empirical studies performed as regards the use of UML diagrams in the maintenance of source code and also on the maintenance of only the UML diagrams themselves. This systematic mapping study covers papers published in journals, conferences and workshops, found via six digital libraries in the period between January 1997 and March 2010.

The systematic manner in which this systematic mapping study was carried out, by following the guidelines provided in (Kitchenham and Charters, 2007), makes this study rigorous and fair.

We would like to highlight two problems that were dealt with during the process of the systematic mapping study:

- It is not usually possible to judge the relevance of a study from a review of the abstract alone. The standard of IT and software engineering abstracts is too poor to rely on when selecting primary studies, and this makes it necessary to review the full text. When used properly, structured abstracts are very useful in improving the quality and usefulness of the abstract (Budgen et al., 2011b). Structured abstracts must contain the following sections: 1) Context (the importance and relevance of

the research), 2) Objectives (the main objectives pursued), 3) Methods (the research method followed and the proposal provided to attain the objectives), and 4) Results (the main findings and conclusions obtained).

- The search engines have some limitations when performing the search on the abstract alone, or when the search string is quite complex, and could not therefore be searched directly. The search string thus had to be tailored to each digital library by splitting the original and combining the results manually. Current search engines are not designed to support systematic literature reviews. Unlike medical researchers, software engineering researchers need to perform resource-dependent searches.

During this systematic mapping study we attempted to answer one main research question: *What is the current existing empirical evidence with regard to the use of UML diagrams in source code maintenance and the maintenance of the UML diagrams themselves?*

We found only two papers ([P5] and [P8]) which were able to help us to answer the first part of this question. Two controlled experiments in [P5] report how the presence of UML diagrams can help to reduce the time needed to maintain the source code. These two experiments and the experiment presented in [P8] show that the quality of the modifications made by subjects is greater when UML diagrams are available. Although the existing studies related to the use of UML diagrams in source code maintenance are in favour of using the UML for this kind of tasks since the quality of the modifications is greater when these diagrams are available, few papers concerning this issue have been published.

If we focus on the papers which deal with the maintenance of only the UML diagrams themselves, we detected some studies which present empirical results concerning the benefits of using UML diagrams as opposed to simply using text, or how the availability of some specific diagrams (class, sequence, state, activity and use case diagrams) can be a positive factor in the maintenance of source code. We also found several pieces of research concerning the maintenance of the UML diagrams themselves that reported some factors which can improve that maintenance of these diagrams (such as the use of stereotypes, the use of composite states, the use of a correct level of detail or of a correct layout), and which will eventually influence the maintenance of the software system. We also found studies concerning how factors that are external to the system under maintenance might influence its maintenance, such as the maintainers' experience and ability.

The main findings according to the categories used to classify the 38 selected primary studies are:

- Research method: Most of the studies present the results of controlled experiments.
- Context: Most of the experiments are carried out in a laboratory context.
- Subjects: Most of the experiments are performed by Computer Science undergraduates.

- Dependent variable: The most common dependent variable used in the empirical studies is the maintainability of class diagrams which is usually measured using time and accuracy.
- Available diagrams: The most widely-used diagrams in the studies selected are class and sequence diagrams.
- Object to maintain: Most of the studies focus on maintaining only the diagrams.
- Type of system: Synthetic systems are those most often used in the studies found.
- Origin of diagrams: Most of the studies found use diagrams that are not obtained from a reverse engineering process.
- Treatments: Most of the empirical studies compare different aspects of UML diagrams, for example diagrams with stereotypes vs. those without them, different levels of detail in the diagrams, etc.
- Quality of papers: Almost 90% of the studies have a medium or high quality.

Almost all the studies with regard to the study of the maintenance of the UML diagrams themselves found are experiments that compare different aspects of UML diagrams, but their external validity, i.e., their generalizability, is questionable given the material, tasks and subjects used.

In summary, one of the main findings is that there is a need for studies that take into account the measurement of cost and productivity, which are variables that have great repercussions in industrial contexts. In order to strengthen the external validity, i.e., the generalizability of the empirical results, we suggest that more experiments and case studies should be carried out in industrial contexts, with real systems and maintenance tasks performed by practitioners under real conditions. Studies concerning how to improve the understandability of a UML diagram (and hence the maintainability of the source code) are carried out from different points of view, comparing different variables. In addition, the maintenance of both diagrams and diagrams and code together must be considered in future empirical studies. It is important to note the lack of empirical studies under real conditions, owing to the fact that the majority of the studies presented used toy systems or prototypes. Due to those reasons, we wish to stress the need for further empirical studies carried out in industrial contexts to investigate whether the use of the UML can lead to important differences that make the costs involved worthwhile, particularly as regards source code maintenance.

We suggest that the Software Engineering community should share or exchange available resources, i.e., models and code, using existing repositories (for example, ReMoDD (France et al., 2006)). After collecting the documentation of some systems and selecting the most representative ones, a benchmark could be created in order to make the results of future empirical studies directly comparable. A repository with experimental material would also help researchers to provide more empirical results by generating new studies or replicating the existing ones.

While conducting this systematic mapping study we detected some studies which present empirical results concerning the benefits of using UML diagrams (activity, class, sequence, statechart and use case diagrams) as opposed to simply using text, or

on how the availability of some specific diagrams can be a positive factor in the maintenance of source code. We also discovered several pieces of research concerning the maintenance of the UML diagrams themselves which report some factors that can improve the maintenance of the system.

We would also like to provide references to some of the papers which obtained high scores in the quality assessment (about 30 out of 40 points) and which could be used as examples of good experiments: P8, P10, P14, and P20.

To conclude this paper, we trust that the systematic mapping study published herein will serve both as a guide to past research in the area, and as a foundation for future research. This work is also an attempt to support other researchers and practitioners by providing a library of papers on empirical evidence concerning the use of UML diagrams in the maintenance of both source code and the UML diagrams themselves.

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), and GEODAS-BC project (Ministerio de Economía y Competitividad and Fondo Europeo de Desarrollo Regional FEDER, TIN2012-37493-C03-01).