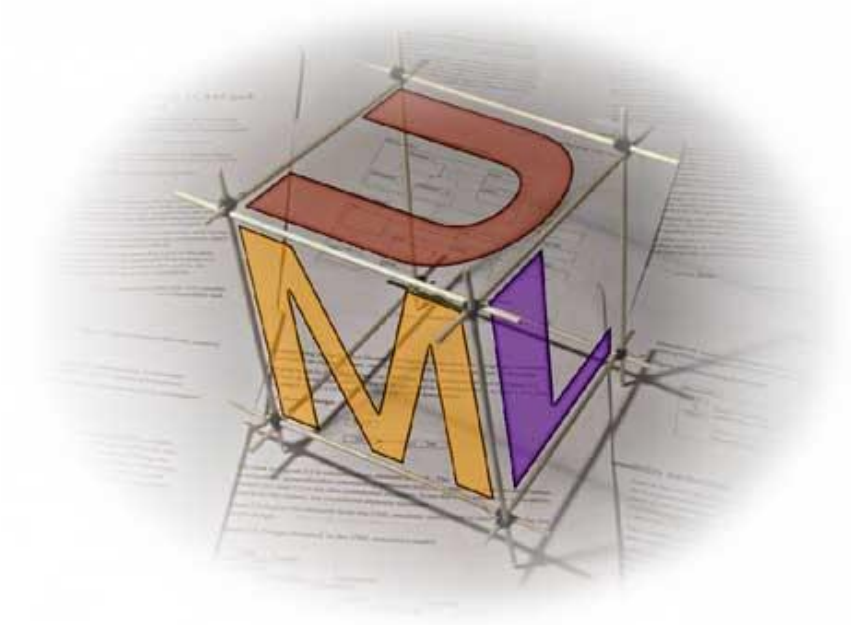
Studying the benefits of using UML on software maintenance : an evidence-based approach

Fernandez Saez, A.M.

CHAPTER 1. INTRODUCTION



1.1. INTRODUCTION

This PhD thesis presents a collection of published papers. This chapter is therefore devoted to presenting the relevance of the theme of this thesis and the problem statement which motivated us to research it. Our research objectives are also described, along with an explanation of the research methods employed and the context of this dissertation. Finally, the structure of this thesis is outlined and the main contributions are summarised.

1.1.1. Relevance of modeling in software projects

The increasing complexity of software projects (van Vliet, 2008) has led to the emergence of modelling languages with which to increase the understanding between customer and developer (in the analysis phase), improve communication among team members (Nugroho and Chaudron, 2009), and increase the understanding of how software works (both in the development and the maintenance phases). In practice, the quality of software models is therefore primarily important for software architects, developers, and maintainers. Software architects, for example, need to ensure that high level design decisions are appropriately translated into detailed designs. Software developers need comprehensible and consistent models on the basis of which the implementation is constructed. Finally, software maintainers depend on models that have a sufficient correspondence with the actual implementation in order for them to perform maintenance activities efficiently.

Modelling is an activity that is carried out to create representations of the software systems, which involves creating technical solutions that meet the requirements of a system. Good software models perfectly capture user requirements and translate them into technical designs. However, a perfect match to requirements is not the only aspect that is crucial for software models. Models of systems are also characterised by their syntactic and semantic quality. Paying attention to the quality of software models is important because it might determine the quality of the final software product. This is particularly true when software models are used as a foundation on which to develop the actual software system.

In order to increase the quality of the final software product, it is important to pay attention to the software representation, and it is therefore advisable for the software representations to conform to a standard specification of a modelling language. The Unified Modeling Language (UML) originated in 1997 in order to be used as a graphical modelling language geared towards modelling object-oriented software systems (Blaha and Rumbaugh, 2004). It was proposed as a standard for the Object Media Group (OMG) some years later (ISO/IEC, 2005). The current version of UML, version 2.4.1 (OMG, 2011), has 14 diagram types that are divided into two categories, namely structural and behavioral diagrams. UML has been accepted in industry to such an extent that it is now considered as an industrial standard for software modelling (Erickson and Siau, 2007; Grossman et al., 2005). Of the 14 possible diagrams, those most commonly used are the following (Agner et al., 2013; Dobing and Parsons, 2006; Grossman et al., 2005; Hutchinson et al., 2014):

- The class diagram (Figure 1.1): A diagram that outlines the relations between the classes (entities) in a system. This diagram is widely used because it describes the structural aspect of a system. It shows how classes in the system are interrelated, and their static relations. It also captures constraints regarding the way in which classes interact with each other. Depending on their level of detail, this type of diagram may additionally show attributes and operations of classes.

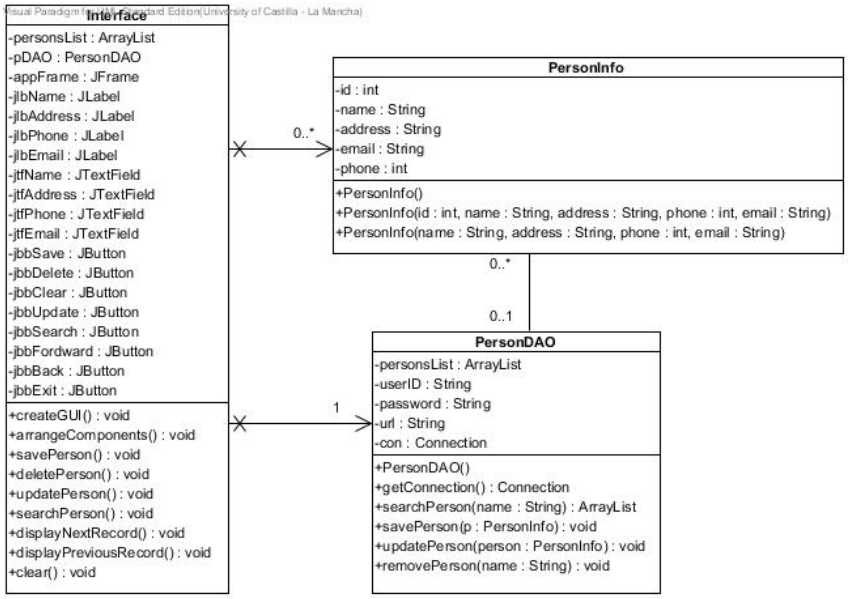


Figure 1.1. An example of a UML class diagram.

- The use case diagram (Figure 1.2): A diagram that depicts an overview of a system in terms of the relation between various usages of a system. This diagram is widely used because it captures the functionality that needs to be delivered by a system. This functionality is visualised in terms of horizontal ellipses that are connected to actors that perform or execute the functionality.
- The sequence diagram (Figure 1.3): A behavior diagram in which a sequence of messages between instantiations of objects is modelled. This diagram is widely used because it captures interactions of objects in order to deliver a particular functionality.

These three diagrams provide three different points of view of the system modelled. These views complement each other, offering different levels of detail or providing different purposes for each view.

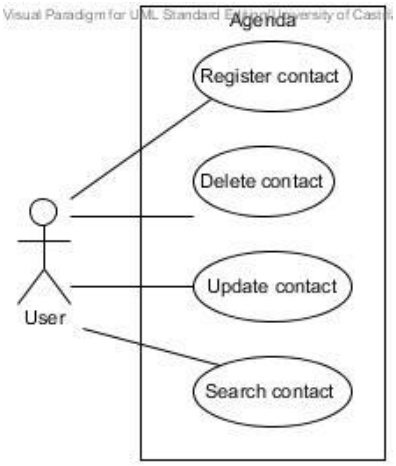


Figure 1.2. An example of a UML use case diagram.

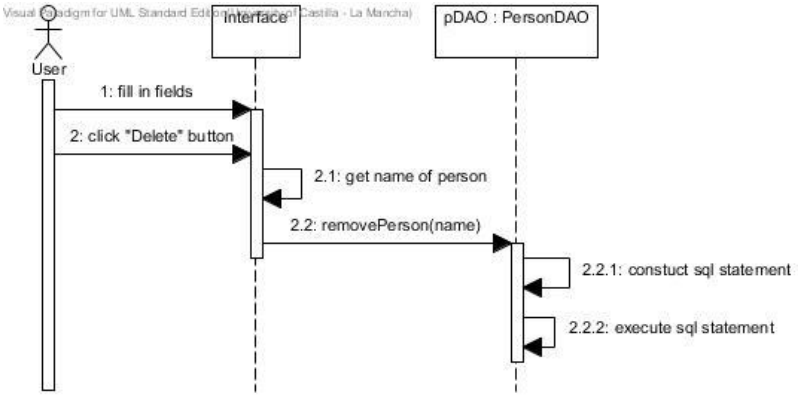


Figure 1.3. An example of a UML sequence diagram.

In practice, UML is used in a variety of ways. Ordered from informal to formal in the sense of diagram completeness and adherence to the UML standard, these are:

- As a sketch: Developers can use the notational elements of UML to quickly draw part of a system for comprehension and communication purposes. For sketches, UML elements such as classes and actors might be used in combination with informal or domain-specific constructs (Cherubini et al., 2007).
- For the communication of system design: Modelling parts of a system makes it possible to explain, for example, how a system component is supposed to function. Depending on how much of a system is modelled, this approach can be a form of model-centric development.
- As a blueprint: In this case, most of the system analysis and design has already taken place and the resulting set of UML diagrams is then used for implementation. This type of development approach is referred to as model-centric development. UML is likely to be used as a blueprint in the context of Global Software Development during which design and coding activities take place at different

geographical locations and there is a need for agreement on the details of the system.

- As a programming language: UML diagrams can be used as a basis for code generation. The UML diagrams must strictly adhere to a predefined syntax. This type of development approach is referred to as Model-Driven Development.

1.1.2. Problem statement

Including modelling as part of software development appears to have various benefits. Why then is it that not all companies use software modelling? One of the main reasons is that it requires up-front investments. From an economic point of view, any type of investment must be justified in terms of how much payback there will be at a later stage. This being the case, in the context of software projects, investment in modelling should be justified by benefits, such as improved productivity and improved product quality, which can be seen later during software development or maintenance. When such benefits are not tangible or foreseeable, modelling becomes a practice without clear added value for the system being developed.

The benefits of using software documentation to comprehend and modify source code have been widely studied (Abbes et al., 2011; de Souza et al., 2005; Tilley and Huang, 2003). The authors of (Tryggeseth, 1997) report that having documentation available during system maintenance reduces the time needed to understand how to perform maintenance tasks by approximately 20 percent. The availability of good documentation is also believed to minimise the loss of information and misinterpretation as regards communicating decisions made during the complete software life cycle. Software documentation might also increase productivity (Arisholm and Briand, 2006).

The big issue regarding software modelling in fact resides exactly in the assumption that it will provide software projects with quantifiable benefits. The problem, therefore, is how we can investigate and prove whether or not modelling, or some specific characteristics of modelling, provide any benefits during software development and maintenance. As long as this question remains unanswered, it will be difficult to motivate and justify modelling activities in real software projects. This thesis therefore contributes to partially answering these open questions by focusing the research on the benefits of using UML modelling during software maintenance. We have focused on the UML as a modelling language because, as mentioned previously, UML has become the de facto standard graphical modelling notation used in software documentation. We have also focused on the maintenance phase because it is the phase of the software life cycle which consumes the majority of software development resources. As explained in (Pressman, 2005) and (Glass, 2002): “Maintenance typically consumes 40 percent to 80 percent of software costs. Therefore, it is probably the most important life cycle phase of software” and “60 percent of the budget is spent on software maintenance, and 60 percent of this maintenance is to enhance”. Enhancing old software is, therefore, a huge undertaking.

1.1.3. Objective of the Thesis

The main goal of our research is to investigate the benefits of modelling in software maintenance. We particularly focus our attention on UML as a modelling language since, as explained previously, it is widely used in industry.

Our main research question is, therefore:

"What is the impact of using UML in software maintenance?"

This main research question was used as the basis to begin reviewing the existing literature (Figure 1.4) in order to discover and analyse what has already been done in this field and to identify the remaining gaps which are potential areas for further investigation.

As the main research question is too general, and after considering the main findings obtained through an exhaustive literature review, three research questions were initially formulated (Figure 1.4):

- RQ1. What is the perception of professionals in industry as regards the value of using UML in software maintenance?
- RQ2. Does the level of detail (LoD) of UML diagrams influence software maintenance?
- RQ3. What is the impact of using UML in software maintenance in terms of its use in a software project?

Moreover, new insights obtained during this long-term study led us to define a new research question:

- RQ4. Are forward designed or reverse-engineered UML diagrams more helpful for code maintenance?

These research questions were formulated in order to observe the impact of UML modeling on software maintenance from different perspectives and in different contexts, such as the academic context, by collecting evidence in a controlled environment from the point of view of students who will be the future professionals, and in an industrial context by collecting data from the point of view of current software engineers.

Bearing in mind that they cover a wide spectrum of research, the questions formulated were addressed by following a mixed-method approach (Creswell, 2013). We were of the opinion that considering different perspectives and combining multiple research methods in order to answer the principal question would enable us to obtain a more comprehensive understanding of the impact of UML modeling on software maintenance.

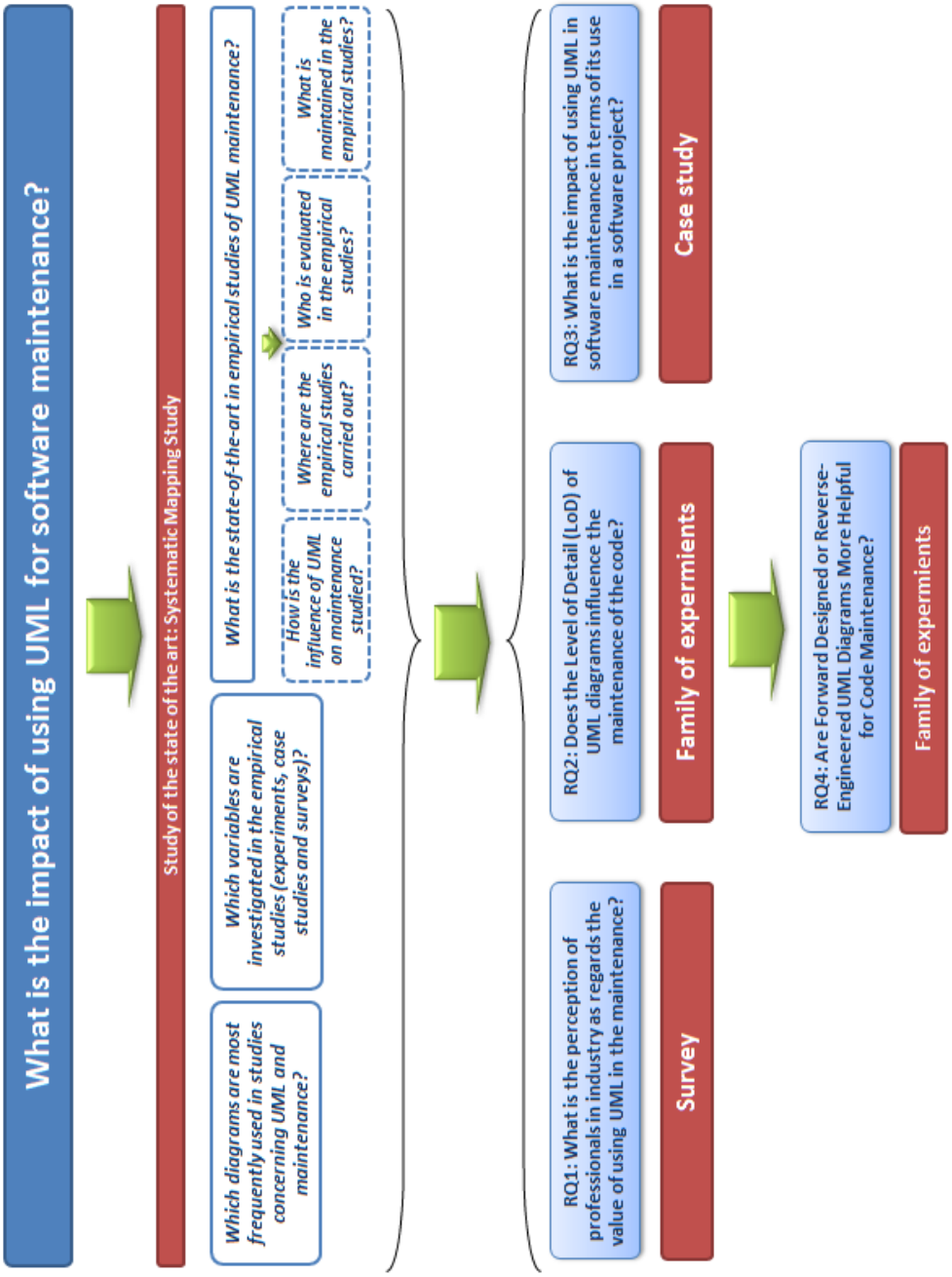


Figure 1.4. Research plan.

1.1.4. Relations between research questions and the main goal

It is important to explain how each research question contributes to obtaining the answer to the main research question:

RQ1. ***What is the perception of professionals in industry as regards the value of using UML in software maintenance?*** This provides a subjective idea of the success of using UML in software maintenance. It also provides insights into how maintainers believe that using UML affects their daily work.

RQ2. ***Does the level of detail (LoD) of UML diagrams influence software maintenance?*** We first considered performing an experiment to discover whether or not maintenance is influenced by the use of UML, but the results of the preliminary study showed us that this topic had already been studied in (Dzidek et al., 2008). While various benefits of using modeling in software development and maintenance are commonly reported, software organizations seem to fear the effort involved in the creation of models are part of documentation. An investment needs a payback. In order not to use more resources for the creation of the UML models than the paybacks that might originate from their use, it would be necessary to define the proper Level of Detail (LoD) of UML diagrams as part of the documentation. To the best of our knowledge, the influence of LoD has already been studied, but in relation to the development (Nugroho, 2009; Nugroho et al., 2008) rather than the maintenance stage. It would be also desirable to help organizations to define the proper LoD necessary to update the UML documentation in order for it to be synchronized with the source code and thus attain understandability benefits during further maintenance. This study, therefore, provides some insights into which LoD in a UML diagram it is best to use in a specific context (novice developers or academic context).

RQ3. ***What is the impact of using UML in software maintenance in terms of its use in a software project?*** This provides an industrial point of view concerning the influence of UML on maintenance tasks. We began performing some interviews related to RQ3, and we found that maintenance tasks are sometimes supported by UML diagrams built at the beginning of the development and that in other cases the diagrams are obtained by means of reverse engineering processes. In order to start our study in an academic context, and whilst waiting to attain an industrial case study that would allow us to work on RQ3, we decided to tackle RQ4.

RQ4. ***Are forward designed or reverse-engineered UML diagrams more helpful for code maintenance?*** As explained previously, this study was carried out in base to results and insights of RQ2 and RQ3, in order to investigate whether the origin of the UML diagrams influences maintenance tasks. Reversed Engineered (RE) diagrams are diagrams with a very high LoD since they represent all the elements in the source code. The Forward Designed (FD) diagrams might also be considered as diagrams with a high LoD depending on their content (class names, attributes, operations and relationships for class diagram; and lifelines, messages and parameters of messages in sequence diagrams). However, FD diagrams more commonly do not represent all the elements in the source code. The key characteristics of the comparison of RE and FD diagrams is: 1) RE diagrams are complete and close to source code; and 2) man-made

FD diagrams are almost never complete, but their strength seems to be the selective inclusion of information about the system and modest complexity of the diagrams. This study presents the results of a family of experiments carried out in an academic context in order to investigate how maintainers work with diagrams generated by means of a Forward-Design process in comparison to diagrams generated using Reverse Engineering techniques.

1.2. RESEARCH METHODOLOGY

As our approach for addressing the research questions in this dissertation is empirical in nature, in this section we shall introduce the main characteristics of the empirical research methods used to obtain empirical evidence that will allow us to answer the proposed research questions (Figure 1.4). A more detailed explanation of them will be provided in the chapter in which each of them is used.

We have selected the most appropriate research method for each research question, signifying that various research methods have therefore been used. This is known as the mixed-method approach (Creswell, 2013). Mixed methods research is an approach that combines quantitative and qualitative research methods in the same research inquiry. Such work can help develop rich insights into various phenomena of interest that cannot be fully understood using only a quantitative or a qualitative method. A method that is applied in isolation has its own strengths but also its own weakness, so a combination of research methods, and particularly a triangulation of qualitative and quantitative data, helps researchers develop a more profound understanding of a phenomenon.

We began collecting the existing empirical evidence on the topic addressed by means of a systematic mapping study; in this case, we followed a combination of the guidelines provided by Petersen et al. (Petersen et al., 2008) and Kitchenham et al. (Kitchenham et al., 2015). We later carried out several empirical studies by considering the following research methods: survey (Fink, 2002; Pfleeger and Kitchenham, 2001), interview (Steinar, 2007) (as a means to perform surveys), families of experiments (Wohlin et al., 2012) and a case study (Runeson et al., 2012).

1.2.1. Systematic mapping study (SMS)

In order to identify the existing research in relation to the topic being studied in this thesis, we first performed an SMS. Systematic Mapping Studies (SMSs) are studies, whose principal objective is to provide a global view of the subject of interest (with or without an empirical approach) and to identify the quantity and type of research available and the results obtained from it. This makes it possible to identify those subjects on which there is little empirical evidence and for which it is necessary to carry out more empirical studies. SMSs usually serve as a basis upon which researchers can carry out future investigations, always supposing that they have been carried out in a rigorous manner (Kitchenham, B. et al., 2011). The well-defined methodology makes it less likely that the results obtained the literature selected will be biased, although it does not protect against publication bias in the primary studies. In

the case of quantitative studies, it is possible to combine data using meta-analytic techniques. This increases the likelihood of detecting real effects that individual smaller studies are unable to detect. The major disadvantage of SMSs is that they require considerably more effort than traditional literature reviews.

SMSs are widely accepted in software engineering (SE), although this form of study appears to be less widely used in other disciplines. This may reflect the nature of SE, which is not a truly "empirical" discipline, although it is slowly moving in that direction. We may, therefore, often have both very limited knowledge about how widely a topic has been studied, and also relatively few studies that are empirical in form. With regard to this, empirical studies may well be reported in many different venues, meaning that we need to carry out a wide search to find all relevant material. As stated in (Kitchenham et al., 2015), an SMS is a useful method for a PhD literature review.

An SMS comprises three principal activities, each of which consists of various sequential tasks (Figure 1.5), of which it is usually necessary to carry out several iterations. A brief description of each of them is provided below:

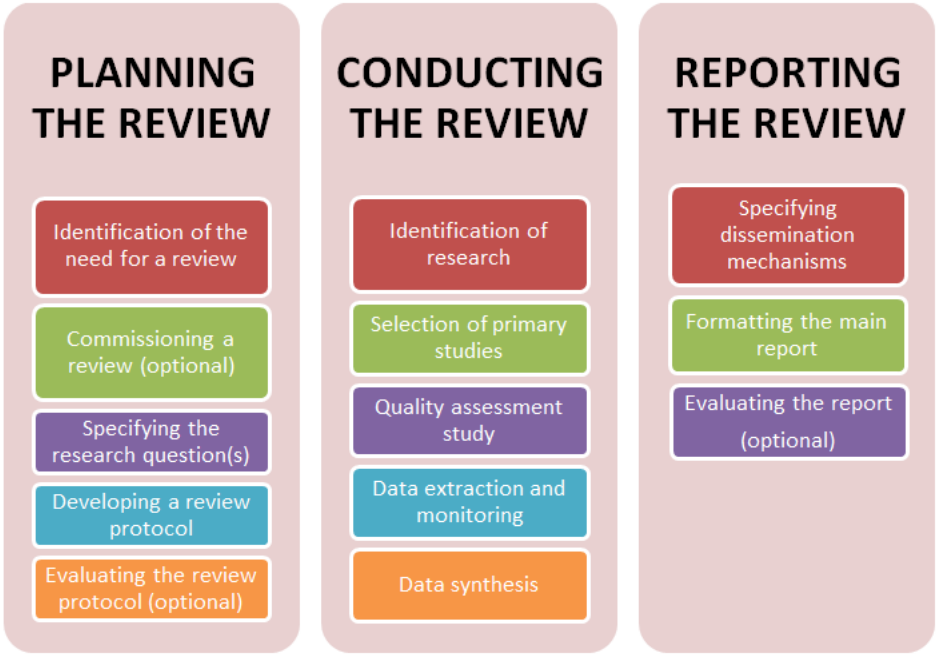


Figure 1.5. Phases of a Systematic Mapping Study.

1. **Planning the review**: Planning is crucial, since the correct development of the SMS will depend on the decisions made during this activity. The principal objective of planning is to specify all those aspects that will make the review systematic and rigorous, avoiding as far as possible all bias and ambiguity and detailing what is denominated as the review protocol.

2. **Conducting the review**: All that was previously planned in the protocol is put into practice in this activity, and the final results that will answer the research questions are obtained. It is also fundamental to document all the incidents that occur and the decisions made during the tasks carried out when executing the review. This will make it possible to replicate the SLR and will signify that all the decisions made will be available for external reviewers and anyone else who might wish to use the results obtained.
3. **Reporting the review**: Finally, a report is written that reflects the entire review process, considering the means used to share the information that was selected when defining the protocol. The recommended structure and content when writing the report of the review are presented in (Kitchenham and Charters, 2007). Space restrictions may make it necessary to complement the publication with a technical report whose website is shown in the report. This makes it possible to include relevant but highly extensive information, such as the complete definition of the protocol, the list of primary studies, the data extraction forms filled out with information on the case study, the evaluation of the quality of each study, etc.

1.2.2. Experiment

An experiment is a formal, rigorous and controlled investigation in which the relation between an effect and a cause is addressed (Wohlin et al., 2012). In software engineering, an experiment is empirical research that handles a variable (denominated as the independent variable or factor) of the environment or phenomenon being studied and measures the effect that it has on another variable denominated as the dependent variable. In a controlled experiment, the effect of a treatment on an experimental group is tested and compared to that of a control group in which the treatment was absent. If the experimental design is in line with the study objective and executed in line with methodological guidelines (as is described in (Wohlin et al., 2012)), the results obtained are regarded as strong evidence. The principal strength of experiments is that they can be used to investigate in which situations statements are correct and may serve to recommend those contexts in which certain standards, methods or tools are useful. One disadvantage of using experiments is that, because of the need to exclude other variables (alternative explanations), the experimental setup has little resemblance to industrial reality. This potentially limits the generalisability of results.

Given that experiments or isolated studies rarely provide sufficient information with which to respond to the questions defined during research, it is of interest to ensure that these experiments form part of a family of studies rather than considering isolated experiments (Basili et al., 1999). These families of experiments may make it possible to extract relevant conclusions concerning the hypotheses that cannot be obtained with individual studies. The studies of which families of experiments are formed should be planned appropriately as replicas of the original experiments.

The synthesis or quantitative addition of the results obtained using families of experiments are continuously carried out by means of meta-analysis techniques (Glass

et al., 1981). Upon combining the results of various experimental studies, the meta-analysis makes it possible to generate more general and reliable knowledge than that of results obtained by means of individual studies, since that knowledge is supported by a greater quantity of empirical evidence.

If all the studies included in the meta-analysis process are equally precise and employ exactly the same independent variables, it is sufficient to average out the results of each one in order to obtain a final conclusion (Borenstein et al., 2007). However, in practice not all studies have the same precision, and when they are combined it is therefore necessary to assign a greater weight to those that allow more reliable information to be obtained. This is achieved by combining the results using a weighted mean (Cochrane Collaboration, 2003). Furthermore, in order to resolve problems related to the non-uniformity of dependent variables, the results of meta-analysis methods are expressed by means of an index denominated as 'effect size', which is an estimator of the size of the relationship between a treatment and a dependent variable(Cochrane Collaboration, 2003) and can be applied to any measurement of the difference between the results in two groups. The objective of a meta-analysis is, therefore, to discover the size of the global effect obtained from the sizes of the effects of each individual study and that will reflect the degree to which the phenomenon being studied is present in the population as a whole.

In order to carry out an empirical study, it is necessary to follow a process which details the activities to be carried out, what must be done and what the inputs and outputs of each activity are. The process (Wohlin et al., 2012) shown in Figure 1.6 is focused on experiments (which is the type of empirical study most frequently used in this thesis), but the same basic activities must be carried out in any type of empirical study. The main difference is how each of the tasks in each activity is carried out, i.e. the design of an experiment. A survey or a case study is different, but all of them must be designed. This means that the basic activities are the same, but each activity must be adapted according to the specific objective of each type of study. The experimental process consists of five activities, each with its corresponding tasks, but this process is not carried out sequentially, as it is not necessary to complete one activity before starting another. In fact, the order of the activities in the process only indicates the order in which each activity should be started. The process is iterative, since in certain cases it is necessary to go back in order to refine a previous activity before continuing with the experiment. It is obviously not possible to go back and refine the objective or the planning once the execution of the experiment has begun.

Each of the activities in the experimental process is briefly described below:

1. **Scope definition**: The objectives of the experiment must be defined in this stage. In order to correctly define all the important aspects of the experiment before going on to its planning and later execution, it is advisable to use the GQM (Goal-Question-Metric) template to define objectives (Basili and Rombach, 1988).

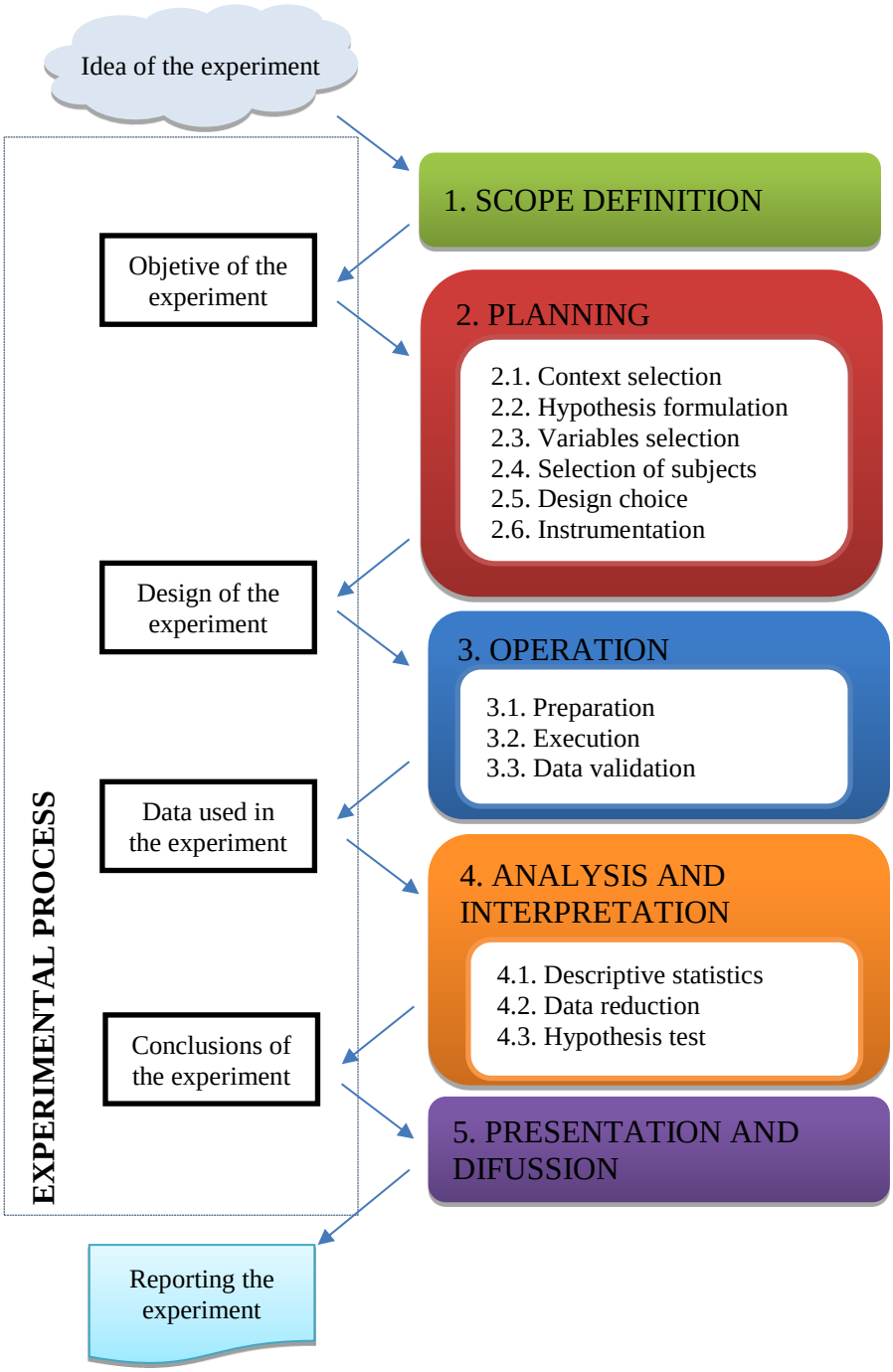


Figure 1.6. Global view of the experiment process.

2. **Planning**: After defining the objectives of the experiment, it is planned in order to attain a clear idea of how it will be carried out.
3. **Operation**: Once the experiment has been planned and designed, it must be carried out in order to collect the data that will later be analysed. This is denominated as the operation of the experiment.
4. **Analysis and interpretation**: After collecting the data obtained once the experiment has been carried out, it is necessary to analyse and interpret them correctly. Three principal aspects should be borne in mind when choosing from among the various analysis techniques: the nature of the data collected, the motive of the experiments and the type of experimental design.
5. **Presentation and diffusion**: When an experiment is carried out, its findings are usually presented, which can be done by communicating it at a congress or in a journal, by means of a report in order to make decisions or as a package that can be used to replicate the experiment, or as educational material. Whatever the case may be, the most important thing is not to forget any important aspects of each of the activities in the experimental process described above. A good documentation of the experiment will allow other researchers to replicate it or use the empirical knowledge contained in it as a basis for other research.

1.2.3. Survey

A survey is a comprehensive system that is used to collect information in order to describe, compare or explain knowledge, attitudes and behavior (Pfleeger and Kitchenham, 2001). Survey methods are a well-established social science technique with which to obtain a broad characterisation of a particular issue (Hutchinson et al., 2014) because, as a data collection method, surveys have several crucial potential advantages over less systematic approaches. When they are properly designed, executed, and described, they (1) economically present the characteristics of a large group of objects or respondents and (2) permit an assessment of the extent to which the objects measured or respondents are likely to adequately represent a relevant group of objects, individuals, or social organisms (Diamond, 2011).

In the majority of cases, the data relative to the survey are obtained by means of questionnaires. But questionnaires do not, in themselves, constitute a survey. In fact, a survey is a more complex process that is formed of a series of well defined activities (Kitchenham and Pfleeger, 2008) which are enumerated as follows and which served as guidelines for the development of our survey:

1. **Establish the objectives of the survey**: establishing the objectives is always the first step when conducting a survey or any other type of research. In the specific case of surveys, each objective must be established as a sentence that is relative to the results expected after carrying out the survey itself.
2. **Design the survey**: the two types of survey design most frequently used are: Transversal surveys, which request information from the participants at a particular moment, and 2) Longitudinal surveys, in which the goal is established in the longer

term and the aim is to discover the evolution of a particular population (which may be the same or may change) over time.

3. **Develop the questionnaire**: the objective of a survey is to obtain responses to a series of questions for a particular reason. When designing a survey it is therefore appropriate to base it on the research objective. However, directly translating objectives into questions rarely leads to a useful questionnaire, and if the objective is to attain a genuinely effective survey it is therefore necessary to design the questionnaires used to collect the data correctly and cautiously. Evaluating and validating the questionnaire: once the questions in the questionnaire have been defined, it is necessary to ensure that they will be understood correctly. This can be done by evaluating the probable index of response, and evaluating the reliability and validity of the questionnaire by means of discussion groups and/or pilot studies, and verifying that the data analysis method used will be compatible with the response that will be obtained.
4. **Obtain the data from the survey**: when obtaining the data, it is normally impossible to attain the responses from the entire population implied in the study, and it is therefore necessary to resort to a sample of it in the hope that the responses will represent those that would have been obtained from the complete set.
5. **Analyse the data obtained**: once the survey has been designed and carried out, it is time to analyse the data obtained from it. It is at this moment that the researchers deal with the most important points regarding data analysis. More specifically, they study the validity of the data obtained, the division of the response into homogeneous groups and, finally, the analysis of both the ordinal and the nominal data obtained.
6. **Report the results**: this step is concerned with publishing the information generated when carrying out the survey and the analysis of its results in a journal or at a congress, or even in the form of a technical report associated with a research project.

1.2.4. Case study

A case study is ‘‘*an empirical enquiry that draws on multiple sources of evidence to investigate one instance (or a small number of instances) of a contemporary software engineering phenomenon within its real-life context, especially when the boundary between phenomenon and context cannot be clearly specified*” (Runeson et al., 2012). Case studies are conducted by collecting qualitative information, which is rich but lacks a standardised structure and analysis methods. The nature of a case study is inimitable, and such studies are thus difficult to repeat with other cases. This nature signifies that case studies are sometimes criticised for being impossible to repeat and generalise from, for being biased by researchers, and so on (Runeson and Höst, 2009). For example, case studies do not generate the same results when they are repeated, unlike analytical and controlled empirical experiments. However, the main advantage of case studies is that they provide a more profound understanding of the phenomena under study. In order to mitigate this problem, we followed the guidelines proposed by (Runeson et al., 2012) in order to design, analyse and report the findings

of the case study presented in this thesis. The most commonly used process is based on the following five activities (Runeson et al., 2012):

1. **Design and plan the case study**: when planning the case study it is necessary to take into consideration at least the following elements: the objective of the study, the case and the unit of analysis, the theory, the research questions, the data collection method and the data selection strategy.
2. **Prepare the data collection**: first, second or third degree data collection techniques are established according to the degree to which the researcher is implied in data collection. Of these, there are interviews, observations, collection of data in files or the calculation of metrics.
3. **Collect the data**.
4. **Analyse and interpret the data collected**: data analysis is carried out differently for qualitative and quantitative data. In the case of qualitative data, an analysis of the descriptive statistics, a correlation analysis, predictive models and a contrast of hypotheses are usually carried out, while in the case of quantitative data, generation techniques and the confirmation of hypotheses are carried out.
5. **Report the results obtained**.

These activities are generally similar to those of other empirical studies, such as the experimental process presented above (see Figure 1.6), although case studies have certain peculiarities.

In order to analyse the data collected as part of the case study presented in this thesis, we used grounded theory (Strauss and Corbin, 1990). Grounded theory (Service, 2009) is an analysis method that is geared towards theory development. Grounded theory is a primary research approach that describes the qualitative sample methods, data collection and analysis, the use of the constant comparison method, the use of theoretical sampling, and the generation of a new theory. It treats the study reports as data that can be analysed in order to generate superior topics and interpretations. The steps that must be followed in order to understand the complete process of grounded theory are (Strauss and Corbin, 1990):

1. The data are collected.
2. The evidence in the data is analysed.
3. If replicas are found in the data, it is possible to start discussing facts.
4. A theory begins to be generated.
5. Concepts that interpret the data are created from the theory.

Interviews were used as a qualitative data collection method as part of the case study. This technique "*seeks to cover both a factual and a meaning level*" (Kvale, 1996). Interviews are used when data needs to be collected about phenomena that cannot be obtained using quantitative measures. The type of interview used for data collection in the context of the case study is the "qualitative interview" which is "a sort of guided conversation" (McNamara, 1999). The interviews are standardised in the sense that each interviewee is asked similar questions (depending on that person's role) and open-ended in the sense that there is ample room for interviewees to elaborate. This type of interview is also referred to as semi-structured or focused.

1.3. CONTEXT OF THE THESIS

The author of this thesis has developed it as part of a co-supervision agreement between the University of Castilla-La Mancha (Spain) and the University of Leiden (The Netherlands), receiving financial support from both of them. She is a member of the Alarcos Research Group (http://alarcos.esi.uclm.es), which is led by Professor Mario Piattini. The author of this thesis joined the group as a research fellow in 2007.

She has also developed part of this thesis in collaboration with the FaST-SE research group at the University of Leiden (The Netherlands) which is led by Michel R.V. Chaudron. During 2012, the author was a research member of the FaST-SE.

During the thesis development, she also participated in a collaboration with the Software Engineering Research Laboratory (SERLAB), led by Giuseppe Visaggio, in the Department of Informatics at the University of Bari, Italy.

Finally, the last collaboration was with the Department of Computer Science and Engineering (CSE) of the Chalmers University of Technology at the University of Gothenburg, Sweden.

The afore mentioned research collaborations were part of the following pre-doctoral stays:

- University of Leiden: There were three stays in this university: The first stage was from 11/06/2010 to 31/01/2011, the second was from 01/01/2012 to 22/05/2012, and finally, the third stage was from 30/05/2012 to 31/12/2012.
- University of Bari: This stay lasted from 23/05/2012 to 29/05/2012.
- Chalmers University: There were 2 stays at this university: 1) From 4/12/2013 to 13/12/2013 and 2) From 19/072013 to 02/08/2013.

During the rest of the time since 01/06/2010, and discounting the periods of stays abroad, the PhD student has developed her research in the Department of Technologies and Information Systems at the University of Castilla-La Mancha.

Leiden University provided the author of this thesis with economical support by employing her as a Research Assistant during the 20 months mentioned above. She was also employed as a Research Assistant at the University of Castilla-La Mancha during the rest of the time needed to complete this thesis. Some of the research included in this thesis has, therefore, been developed in the context of the following research projects (Table 1.1 and Figure 1.7):

Project	Main Researcher	Duration	Participant entities	Founding body	Budget
MEDUSAS: Mejora y Evaluación del Diseño, Usabilidad, Seguridad y Mantenibilidad del Software (Improvement and Evaluation of Design, Usability, Security and Maintainability of Software)	Mario Piattini Velthuis	01/01/2009 - 01/01/2013	University of Castilla-La Mancha, Alarcos Quality Center, S.L.; Sicaman Nuevas Tecnologías, Audisec S.L., Génesis XXI	CDTI-MICINN ((IDI-20090557)	364.700 €
MECCA: Metodología para la Evaluación Continua de la Calidad de Artefactos software (Methodology for Continuous Quality Assessment of Software Artifacts)	Marcela Genero Bocco	01/04/2009 - 31/03/2012	University of Castilla-La Mancha	Junta de Comunidades de Castilla- La Mancha (PII2I09-0075-8394)	200.000 €
MAGO /PEGASO: Mejora Avanzada de Procesos Software Globales (Advanced Improvement of Global Software Process)	Mario Piattini Velthuis	01/10/2009 - 30/12/2012	University of Castilla-La Mancha, University of Murcia in coordinated project (PEGASO: Procesos para la mEjora del desarrollo GlobAl del Software (Processes to improve global software development)	CDTI-MICINN (TIN2009-13718-C02-01)	489.100 €
EECCOO: Entornos para la Evaluación Continua de la Calidad de artefactOs (Environments for Continuous Evaluation of Artifacts Quality)	Marcela Genero Bocco	01/03/2010 -28/02/2012	University of Castilla-La Mancha, Alarcos Quality Center	MICINN (TRA2009-0074)	102.245 €

Table 1.1. Projects that form the context of the thesis (part 1/2).

Project	Main Researcher	Duration	Participant entities	Founding body	Budget
ORIGIN: Organizaciones Inteligentes Globales Innovadoras (Global Innovative Intelligent Organizations)	Mario Piattini Velthuis	06/04/2010 -31/12/2012	University of Castilla-La Mancha, Sicaman Nuevas Tecnologías, EIDOS, INDRA, SIGTEL	CDTI-MICINN and FEDER IDI-2010043(1-5)	7.312.357 €
GEODAS: Gestión para el Desarrollo Global de Software mediante Ingeniería de Negocio y Entornos Avanzados de Colaboración (Management for Software Global Development by Business Engineering and Advanced Collaborative Environments)	Mario Piattini Velthuis	01/01/2012 -31/12/2015	University of Castilla-La Mancha, Sicaman Nuevas Tecnologías, ALHAMBRA- EIDOS, INDRA, SIGTEL	MINECO/FEDER (TIN2012-37493-CO3-01)	308.800 €
IMPACTUM: Aplicación de ISBW en el estudio del IMPACTo de UML en el desarrollo de software en un contexto DSDM-LPS (ISBW application in the study of the impact of UML in software development in a DSDM-LPS context)	José Antonio Cruz-Lemus	29/09/2014-30/09/2015	University of Castilla-La Mancha	Junta de Comunidades de Castilla- La Mancha (PEII11-0330-4414)	50.000 €
SEQUOIA: Security and QUality in processes with bIg data and Analyitics	Marcela Genero Bocco and Eduardo Fernández-Medina Patón	01/01/2016 – 31/12/2018	University of Castilla-La Mancha, University of Alicante, and University of Sevilla	MINECO/FEDER (TIN2015-63502-C3-1-R)	110.100 €

Table 1.1. Projects that form the context of the thesis (part 2/2).

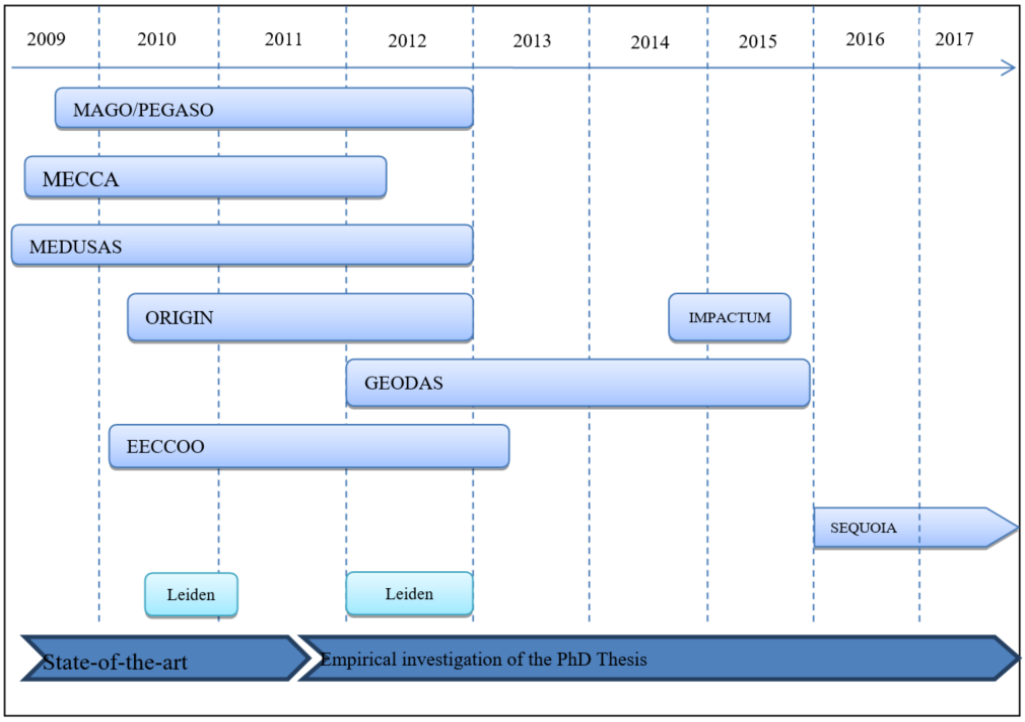


Figure 1.7. Research project timeline during PhD thesis development.

1.4. CONTRIBUTIONS OF THIS THESIS

In essence, the contribution of this study is two-fold. First, this study provides sound empirical evidence about the potential benefits of UML modeling in software maintenance. The findings of this study should also contribute to the body of knowledge, particularly in the field of software engineering. Additionally, from a research perspective, we consider this study as a milestone towards a more comprehensive understanding about the role of modeling in software maintenance.

The second contribution is related to modeling practices. This study provides recommendations concerning best practices of modeling using UML. These recommendations are based on empirical data from an industrial case study as well as expert opinions. Therefore, this study essentially serves as a guide and justification for software engineers to continue using (and updating) the UML models of software systems, and to get the most benefits out of it.

1.5. OUTLINE OF THE STRUCTURE OF THIS THESIS

In this thesis, we have decided to present the results which were obtained from the empirical studies starting with those which took place in academic contexts followed by results obtained from industry. We decided to follow this approach rather than presenting the results in the order of their date of execution, or in the order of the

number of the research question so as to move from less generalizable results to more real ones.

This dissertation is presented as a compendium of papers. Most of the chapters are based on one or more published research papers in order to cover a research question, but only the most complete paper is presented as a chapter in this dissertation.

A summary of the chronology of these published papers and the stays that took place during this research is presented in Figure 1.8:

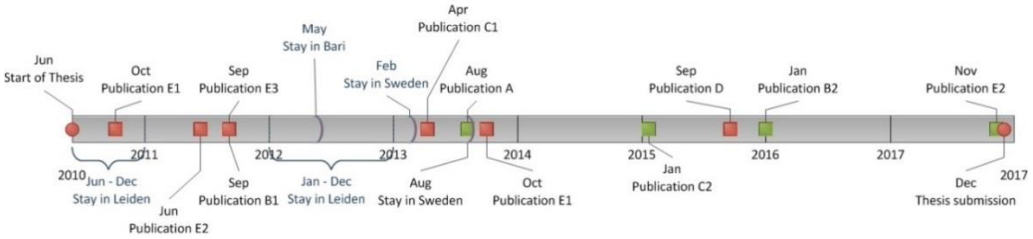


Figure 1.8. Chronology of publications and stays of this PhD Thesis.

Five of the published/under submission research papers have been selected to be part of the core chapters of this PhD thesis owing to both their scientific contribution and their relevance. They are either papers that are indexed in the *Journal of Citation Report* or that appeared in congresses or workshops relevant to the sphere of software modelling.

Bearing the selected papers in mind, the content of what remains of this document is organised as follows:

Chapter 2: State of the Art. This chapter contains the Systematic Mapping Study of the relevant literature, i.e. it provides an exhaustive review of the existing literature, which is the background to the thesis. This chapter is based on the following research paper:

> A) 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**).

Chapter 3: RQ2. Does the level of detail (LoD) of UML diagrams influence software maintenance? This chapter summarises the result of a family of experiments performed with the aim of investigating what different levels of detail in a UML representation would influence the tasks performed by software maintainers. In addition to using objective measures applied to the understandability or modifiability of the system being maintained (such as efficiency or effectiveness), subjective perceptions were also obtained.

Two research papers were written on the basis of the research developed to answer RQ2.The first contains the initial experiment of the family of experiments and was published as a conference paper. The second contains the complete family of

experiments and was published as a journal paper. The second, which is the most complete and recent paper, was therefore selected to be presented in this document:

> B1) Fernández-Sáez, A. M., Genero, M., and Chaudron, M. R. V. (2011). Does the Level of Detail of UML Models Affect the Maintainability of Source Code? Experiences and Empirical Studies in Software Modelling (EESSMod'2011), Workshop at MODELS'2011, LNCS 7167, pp. 133–147.Wellington, New Zealand.
>
> B2) Fernández-Sáez, A. M., Genero, M., Chaudron, M. R. V., Caivano, D. (2016). Does the level of detail of UML diagrams affect the maintainability of source code?: a family of experiments. *Empirical Software Engineering.* 19(6) pp. 1-48 (**JCR Index 2016 = 3.275**)

Chapter 4: RQ4.Are forward designed or reverse-engineered UML diagrams more helpful for code maintenance? This chapter summarises the results of another family of experiments whose objective was to investigate how maintainers perform their maintenance tasks when using UML diagrams originating from a forward design approach (i.e. they are human-based diagrams) in comparison with the usage of UML diagrams generated by Reverse Engineering tools (i.e. they are automatic diagrams). Objective and subjective measures were again used in order to obtain a broader result.

Two research papers were written on the basis of the research developed to answer RQ4. The first contains the initial experiment of the family of experiments and was published as a conference paper. The second contains the complete family of experiments and was published as a journal paper. The second, which is the most complete and recent, was therefore selected to be presented in this document:

> C1) Fernández-Sáez, A. M., Genero, M., Chaudron, M. R. V, Ramos, I. (2013). Are Forward Designed or Reverse-Engineered UML Diagrams More Helpful for Code Maintenance?: A Controlled Experiment. *International Conference on Evaluation and Assessment in Software Engineering (EASE'2013*), pp. 60-71, Porto de Galinhas, Brazil (one of the three best paper awards).
>
> C2) Fernández-Sáez, A. M., Genero, M., Chaudron, M. R. V, Caivano, D., Ramos, I. (2015). Are Forward Designed or Reverse-Engineered UML Diagrams More Helpful for Code Maintenance?: A Family of Experiments. *Information and Software Technology*. 57(1) pp. 644–663, (**JCR Index 2015 = 1.569**).

Chapter 5: RQ1.What is the perception of professionals in industry as regards to the value of using UML in software maintenance? This chapter presents the results of a survey with industrial professionals involved in software maintenance projects. They provided their opinion on how documentation (containing or not containing UML diagrams) enables them to perform their maintenance tasks, among other issues. The results of this survey also reflect what kinds of companies are fruitfully using UML diagrams as part of their software documentation.

One paper appeared as a result of the research related to RQ1, which is included in this thesis:

> D) Fernández-Sáez, Ana M., Caivano, D., Genero, M., and Chaudron, M.R.V (2015). On the Use of UML Documentation in Software Maintenance: Results from a Survey in Industry. Proceedings of the ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS'2015), pp. 292-301, Ottawa, Canada.

Chapter 6: RQ3. What is the impact of using UML in software maintenance in terms of its effectiveness when used in a software project? This chapter provides an industrial point of view as regards how using or not using UML diagrams as part of the software maintenance documentation might provide a return on the investment made during the design phase of a software life cycle. This study was performed as a case study in an industrial context: the software department of a multinational company. This case study consists of a quantitative analysis of the data extracted from the company repository and a qualitative analysis based on a series of interviews.

Two research papers were published on the basis of this case study, although only the latter is included in this thesis. The first contains the initial results of the case study and it was published as a conference paper. The second contains the complete results of the case study and was sent to a journal in order to be published. The second paper, which is more complete and recent was selected to be presented in this document:

> E1) Fernández-Sáez, A. M., Chaudron, M. R. V., Genero, M. (2013). Exploring Costs and Benefits of Using UML on Maintenance: Preliminary Findings of a Case Study in a Large IT Department. Proceedings of the Experiences and Empirical Studies in Software Modelling Workshop (EESSMod'13) at MODELS 2013, pp. 32-42, Miami, Florida, USA.
>
> E2) Fernández-Sáez, A. M., Chaudron, M. R. V., and Genero, M. (2018). An industrial case study on UML modelling practices and their effectiveness in software maintenance. *Empirical Software Engineering* (**JCR Index 2016 = 3.275;** published online 16th March 2018).

Chapter 7: Conclusions. The last chapter of this thesis summarises the results obtained thanks to the different empirical studies carried out in the previous chapters, with the goal of answering the six research questions formulated. Moreover, the main contributions are highlighted and future work is also outlined.

The following papers were also published in the context of this dissertation, and are related to the current chapter, the introduction. The motivation to undertake the studies shown in this thesis and the research plan are presented in them:

> F1) Fernández-Sáez, A. M., Genero, M., and Chaudron, M.R.V. (2010). Investigating the benefits of UML on software maintenance: A research proposal. Proceedings of the 9th edition of the BElgian-NEtherlands software eVOLution seminar (BENEVOL 2010), pp. 34-39. Lille, France.
>
> F2) Fernández-Sáez, A. M., Genero, M., and Chaudron, M.R.V. (2011). Empirical investigation on the benefits of using UML in software

maintenance. Proceedings of the 12th International Conference on Product Focused Software Development and Process Improvement (PROFES'11) – Doctoral Symposium, pp. 44-47. Torre Cane, Italy.

F3) Fernández-Sáez, A. M., Genero, M., and Chaudron, M.R.V. (2011). A research plan for gathering empirical evidence of the benefits of using UML in software maintenance. Proceedings of the 16th Jornadas de Ingeniería del Software y Bases de Datos (JISBD'11), pp. 511-518. La Coruña, Spain.