SUMMARY

Nowadays, it seems that companies do not use software modelling because they fear it requires up-front investments. From an economic point of view, any investment must be justified in terms of how much payback there will be at a later stage. 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 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 provides 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.

Our approach for addressing the research questions in this dissertation is empirical in nature. The selected research methods are: a survey, a case study (including interviews) and two families of experiments. We used mixed methods research as an approach to combine quantitative and qualitative research in the same research inquiry.

Our experiments were carried out with students, whilst the survey and case study were done with professionals, to obtain realistic opinions from industrial context as regards how UML diagrams (and documentation in general) enable them to perform their maintenance tasks.

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. As results of this evidence-based research we could highlight the following findings:

- The use of a graphical notation as a complement when attempting to understand the system that will be maintained and to support the design of the changes related to software maintenance tasks is widespread (especially UML).
- The UML adoption is directly influenced by the educational level and the experience in ICT of the maintainer, the size of the maintenance team and the size of the system under maintenance.
- UML modeling is commonly used at a low level of formality, given that the main purpose of UML models is communication and getting an overview of the system. Another reason is that models in project documentation should be understandable for all the audiences of the documentation.

- Maintainers do not always use the available documentation (with or without diagrams), because use only the source code and its comments. This is owing to problems concerning a lack of synchronization caused by non-updated documentation/diagrams.
- When no UML diagrams are available from the initial software development, they are rarely created during maintenance. Performing reverse engineering techniques is difficult, despite the fact that there are automatic reverse engineering tools, but a manual process is required to "clean" the diagrams. Forward designed (FD) diagrams (created during software development phase) to some extent, improve the maintenance of source code. Only in the case of junior maintainers was a better efficiency obtained when using the Reverse Engineered (RE) diagrams. This might be explained by the fact that Reversed engineered diagrams have a very high traceability with source code, so inexperienced maintainers would prefer this kind of diagrams. In the rest of the cases, Forward Designed class diagrams provide a more attractive balance between detail and relevant information than Reversed Engineered class diagrams.
- The preferred Level of Detail (LoD) of diagrams depends on their purpose: there is a slight tendency in favor of using high LoD diagrams to understand the system, but there is a preference for a low LoD to maintain it. Despite the fact that the participants stated that they experience more difficulties when reading high LoD diagrams, the majority of them considered that the presence of some elements represented in the high LoD versions of each diagram is very valuable: attributes and operations in class diagrams and formal messages with parameters in sequence diagrams.
- Also, several perceived benefits of using UML during software maintenance have been found during this research:
  - Using UML less time is needed for a better understanding of the system being maintained; this improves the detection of defects.
  - Also the reduction in the execution time of a project is considered as a benefit of UML usage.
  - Moreover, when UML diagrams are available as part of the documentation, less effort is required to consult it.
  - If we focus on the product quality benefits of using UML modeling, the majority of maintainers believe that it improves the quality of the ultimate software product.
  - Some process benefits were also highlighted by maintainers involved in this research, such as: improved communication, the prevention of knowledge evaporation and the easing of the diagnosis of problems (especially when using behavior models). UML modeling is also considered helpful as regards improving the design before starting implementation activities (through an increased ease of peer-reviewing).

The second contribution of this Thesis 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 to continue using (and updating) the UML models of software systems, and to get the most benefits out of it.

Finally, we should stress that the benefits and costs/hurdles are difficult to quantify because they are intangible: no company keeps a record of miscommunication, poor early design choices, out-of-date or unavailable documentation and their associated costs. One of the reasons why such benefits are hard to quantify is that modeling is often one of many ways (employed in conjunction with others) of achieving a particular goal. Faults cannot, therefore, then be traced back to a single cause.

The findings obtained are a first approach to discovering the contextual factors that affect the effective use of UML during software maintenance in industrial projects. This research opens up directions for future research. For example, it would be interesting to analyze the effect of different UML notations, comparing formal diagrams with box & lines, or developing supporting tools for a proper UML documentation.

The results of this Thesis contribute to the Software Engineering body of knowledge, allowing to practitioners to have evidence to defend and promote the use of (UML) modeling in software maintenance. The recommendations provided in this Thesis can be applied in real projects of software maintenance companies, improving the daily work of their maintainers and the quality of their projects.