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

Fernandez Saez, A.M.

**Propositions**

*Stellingen*

Ana M. Fernández Sáez, Author of

**Studying the Benefits of Using UML Software Maintenance: an Evidence-Based Approach**

1. *The value of documentation is in the eye of the beholder*: Experienced maintainers of software systems do not need very detailed information about the system when performing maintenance tasks on small systems from well-known domains and they, therefore, prefer to work with low level of detail UML diagrams. But in the case of novice maintainers modifying that kind of systems, using high level of detail UML diagrams improves their understanding of the system and source code. [Chapter 3]
2. In spite of common beliefs, software documentation is of great value. The fact that programmers use UML diagrams when performing software maintenance tasks implies that the investment made by architects when creating UML diagrams has a payback not only as regards the initial software development but also during software maintenance: the maintainer's understanding improves. [Chapter 5]
3. Despite the fact that much research has been carried out into software versioning, the issue of 'currency' has not been addressed: If it is unclear whether software-documentation or software models are current during software maintenance, then they are no longer reliable and useful for developers. [Chapter 6]
4. More than twenty years have passed since the introduction of UML as a modelling language, and it would be reasonable to think that the tooling for software modelling would now be mature. However, there is still a lack of tooling that effectively integrates models with the other (text-based) documentation of a software project [Chapter 6]. This is a hurdle that prevents the effective integration of UML models into the overall development process [Chapter7].
5. In terms of effort, maintenance is more important than development in software projects: Hence, in the context of software projects, people underestimate the volume of software maintenance and the time required to carry it out. It is, therefore, important to wait until that phase in order to obtain all the paybacks of software modelling.
6. Software Engineering researchers should conduct more empirical research and integrate their results in order to construct more comprehensive theories about Software Engineering. In comparison to other more concrete engineering disciplines, the acceptance of the modelling of design is challenged by the fact that stakeholders do not recognize the worth of such models when compared to that of source code.
7. A common mistake in software development is the failure to define and communicate who is responsible for maintaining the documentation and models of a software project.
8. Companies sometimes offer to develop software systems at a very low cost in order to acquire development projects. However, the cost of system development to customers is ultimately not low, because the inevitable cost of maintaining the system is not taken into account.
9. When models are not available from the development phase, Reverse Engineering is expected to be an easy approach by which to obtain models directly from source code. However, after carrying out the automatic extraction process, a manual step is needed, which is considered difficult and is often unable to reconstruct the significant design decisions.