Requirement Engineering Process in AMIDST

The handsome AMIDST guys et. al.

Latest version, July 3, 2014

1 Introduction

Despite the fact that the field of requirement engineering RE has existed since the seventies, very few surveys have been conducted on RE for very small software projects with less than 10 developers [1]. In 2007, a study of seven very small software enterprises in Canada was conducted [2]. And in 2011, a study which involved 24 experienced project managers from 24 different small software companies in Chile was conducted [1]. Based on these two studies, it is clear that RE processes in very small companies are more ad-hoc and that there seem to be little agreement on how to conduct the RE process in practice.

In the literature, several ready-to-use approaches for RE on small projects are attempted. In 2004, Nikula presented a basic RE model baRE as part of his PhD thesis [3]. Based on an argumentation in [1], this method may not be suitable for very small projects, because the method is based on results from a study that involved mostly larger companies [4]. Dorr et. al. presented a set of 36 RE practices for RE improvement for small software projects [5]. However, the six companies that was considered had between 20 and 200 employees, meaning that this research was also biased towards medium sized projects rather than very small.

It is also worth mentioning that a large portion of modern software companies follows Agile methods for software development [6]. Paper [7] outlines how to incorporate requirement gathering in small Agile projects. However, in the Agile methodology, requirement engineering is seen as an ongoing process throughout the whole project, which involves a product owner that continuously renegotiates the requirements. Due to the project management process in EU's Seventh Framework Programme, the AMIDST project is limited from following such a process.

Based on the lack of clarity in practice and consensus in literature on a common RE process for small projects, we decided to identify the characteristics of our project and motivate our choices related to the RE process from this. For instance, the AMIDST project is characterized by the fact that there is a description of work that is agreed upon upfront. The AMIDST RE process must comply with the STREP proposal FP7-

ICT-2013-11, and the software must comply with all deliveries in work package 1 to 8.

Another important characteristic, is that there is a huge widespread of stakeholders in the project and that the software is required to interface with three different softwares from three different companies. Because of this, we chose to focus on the functional requirements, which are the requirements that are the most transparent to the user. As will be discussed later, a use-case driven approach [8] and [9] is taken to achieve this.

The report is outlined as follows. In section 2, the basic principles of requirement engineering are briefly outlined. Section 3 starts by describing the main characteristics of the AMIDST project, before the RE process is outlined. In section 4, we have described the realization of the process so far, before the report is concluded in section 5.

2 Basic principles in requirement engineering

In practice, the RE process ends with a document containing a list with requirements, which are in the form of what a software must do or comply with.

To date there is no common definition of RE. Some definitions focus on elicitation of requirements and therefore the interaction with the user, while others focus on the documentation or the specification. A definition that takes both focuses into account is the IEEE standard given in [10]:

1. *The process of studying user needs to arrive at a definition of system, hardware or software requirements.*
2. *The process of studying and refining system, hardware or software requirements.*

In the context of understanding the RE process, it is worth spending some space on defining the requirement itself. A definition of a requirement is given in IEEE standard [10]:

1. *A condition or capability needed by a user to solve a problem or achieve an objective.*
2. *A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification or other formally imposed document.*
3. *A documented representation of a condition or capability as in 1 or 2.*

This definition has a clear focus on the user, the system/system component and also which contract, standard or specification is needed to be met. Notice, that the requirement is related to *what* a system can do and not *how* it is done.

2.1 Activities involved in requirement engineering

The activities involved in RE vary widely, depending on the type of system being developed and the specific practices of the organization(s) involved [11]. These may include:

- Requirements elicitation
- Requirements analysis and negotiation - checking requirements and resolving stakeholders conflicts
- Requirements specification - documenting the requirements in a requirements document
- Requirements validation - checking that the documented requirements are consistent and meet stakeholders needs
- Requirements management - managing changes to the requirements as the system is developed and put into use

These activities are sometimes presented as chronological stages although, in practice, there is considerable interleaving between them.

Anyone who has a direct or indirect influence on the process is identified as a stakeholder. Stakeholders include end-users that will interact with the system, the developers that will maintain the system, management, domain experts, union representatives etcetera. A challenge in the RE process is therefore to keep a smooth communication between the different stakeholders. The use-case driven approach to RE focuses on simplyfing the communication between the end-users and the developers to improve the overall comminication.

2.2 Use-case driven requirement engineering

It has always been a challenge for the software industry to communicate functionality to the users of a software. Moreover, software engineers are often frustrated, because users often do not know what they want. They only have an idea of what they want. To improve this communication, the use-case driven approach was developed in the nineties. It was first published by Ivar Jacobsen [12] and more modern references are [8] and [9]. A use-case focuses only on the interaction between a user and the system. Requirements are always associated with a use-case. This means that the user is requested to only focus on what he/she wants. This is an advantage, compared to the traditional way where requirements are listed in relation to components and subcomponents in the software. The traditional way often lead to a complexity that the user do not understand. Also, it is more common with requirement duplicates in the traditional approach.

A use-case is a list of steps, typically defining interactions between an actor and a system, to achieve a goal. The actor can be a human or an external system. An overview on

how to write effective use-cases is given in [9], where several templates are given. The use-case providers are asked to provide the use-cases in natural language and for each use-case the following questions are central:

1. Who are the actors involved in the use-case? An actor is either a person or an entity that interacts with the software.
2. What is the main event that initiates the use-case? This could e.g. be an external business event or a system event that causes the use-case to begin. It could also be the initial step in a normal work flow.
3. What are the main user actions and system responses that will take place during the normal execution of the use-case?. This dialog sequence will ultimately lead to accomplishing the goal that is implied by the use-case name and description.
4. How can we evaluate the success of the use-case?

It is also common to group the users, or human actors, within an organization into a small set of user groups. The users within each user group need to have similar roles within their organization and their set of competences are expected to be similar.

To understand the use-case driven approach better, it is useful to distinguish between functional and non functional requirements. Functional requirements are those requirements that are directly related to the interaction between the user and the system. The non functional requirements are more hidden for the user are related to the global overall success. For instance scalability, traceability and testability. When use-cases are provided and functional requirements are identified, it is the requirement engineers role to identify, document and communicate these non functional requirements as well. The use-case driven approach to requirement engineering focuses on revealing the functional requirements together with the users. This improves the communication between the users and the developers, because the focus is on what the users wants and less on how it can be done.

3 The AMIDST requirements engineering process

This section contains a description of the AMIDST RE process. We first describe the main characteristics of the AMIDST project that has influenced and shaped the requirements engineering process. Based on these characteristics we present the AMIDST RE process, which is based on the RE process [] described in Section 2 but tailored to the specific characteristics of AMIDST. Since the focus of the requirements engineering process is on the functionality and documentation of the software products being developed, we will, e.g., not cover process-related requirements.

3.1 Characteristics of the AMIDST project

In this section we identify and describe the key characteristics of the AMIDST project that directly influence the requirements engineering process.

Characteristics one: Pre-specified scope of the project

The AMIDST project is funded by the European Union's Seventh Framework Programme for research, technological development, and demonstration. The overall scope and main developments in the project are therefore defined from the beginning of the project period and documented in the description of work []. This will henceforth be referred to as the DoW framework. More detailed requirements pertaining to the functionality and documentation of the developed software should thus fit within the DoW framework, and their necessity in relation to AMIDST should be justified and demonstrated.

Characteristics two: Partners at different geographical locations

The AMIDST consortium consists of 7 partners/stakeholders, 4 industrial and 3 universities, which are situated in 4 different countries. This diverse consortium composition has at least a two-fold impact on the requirements engineering process.

First of all, although AMIDST targets the industrial stakeholders' common need for processing massive data streams, the more intrinsic aspects of the three industrial domains differ significantly. This, in turn, means that the partners will have different (possibly conflicting) requirements for the system being developed. To ensure that the requirements are comparable across domains and abide to the DoW, a unified formal framework for eliciting system requirements is needed. Such a framework may also provide transparency in the overall requirement engineering process and help prioritize requirements across different domains and thereby help resolve potential conflicts.

Secondly, with the project partners located in different countries, there is a need for a controlled and stringent requirements process in order to limit travel expenditures. This approach is supported by a unified formal requirements engineering framework. Consultancy and discussions in relation to the requirements will primarily be achieved through telecommunication conferences and by physical meetings only secondarily.

Characteristics three: Transfer of domain knowledge between partners

The industrial partners of the AMIDST project come from very different domains: the automotive, energy, and finance industry. To ensure the development, refinement, and completion of the unified formal requirements framework it is necessary with regular and structured communications among the project partners during the requirements engineering process. This not only relates to the specific requirements, but also to the software and user context in which the AMIDST framework should be deployed. The

latter part, in particular, is required for a proper evaluation and validation of the elicited requirements.

Characteristics four: One framework for three different domains

The AMIDST system should define a general framework that can encompass the diverse domains of the three industrial partners. Thus, the format of the unified requirements framework should be sufficiently general and flexible to allow for all relevant requirements to be elicited for the three domains. At the same time the framework should be appropriately structured and formalized enabling a controlled elicitation process (see also Characteristic two) with the requirements specified in a consistent manner making them comparable across domains. In order to also provide a basis for a controlled and balanced system development, the requirements should be linked to relevant project phases, work packages, and tasks. This, in particular, will provide the work package leaders with a clear overview of the requirements that are relevant for the activities in a specific work package.

Characteristics five: Potential refinement of project focus

AMIDST is a research project, where both the industrial and academic partners' understanding of the domains develop as the project unfolds. In order to support a potential refinement of the project's focus and goals, the requirements engineering process should allow for an internal (re)prioritization of the requirements that is transparent across application domains.

3.2 Project phases and AMIDST requirements identification

The overall project duration will be decomposed into different phases, each having distinct requirements. According to [13] a project's life cycle can be divided into three general phases: the design phase, operations phase, and disposal phase. The disposal phase is outside the scope of AMIDST and, thus, will not be considered in the present document. The design and operations phase, however, represent the temporal development of the project, and are initiated by the start of the project and ends with the testing of the deployed system. Each phase can furthermore be described as a collection of distinct stages in the project. The overall process is illustrated in Figure 1.

FiXme: *Should we also discuss that the software is expected to live beyond the project ... ?*

In the design phase general functionality requirements for the system is specified, i.e., what the system should do and support. Figure 1 details key stages inside this phase. The first stage consists of the design of the general framework (models and algorithms) as well as the design and development of the software tools. These stages are primarily related to Work packages 1–5. In the second stage, the general framework and software is instantiated for each specific use case. Finally, initial tests of the use case instantiated frameworks are conducted. During this phase of the project, possible design requirements

FiXme: *Describe the content of the two phases, design and operations*

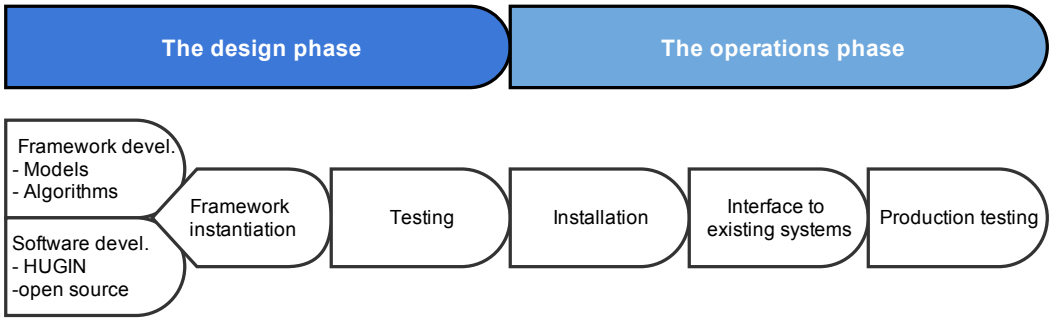

Figure 1: The figure shows the key stages in the design and operations phases. For the requirement specification, each requirement should be defined relative to one of these stages.

could, e.g., address

- the scope of the model
- the interpretability of the learned models
- the extent and type of domain knowledge that can be integrated into the models
- prediction accuracy of the developed models
- documentation

The requirements for the operation phase concern the functionality of the deployed system. In Figure 1, we decompose this phase into three stages: installation, interface to existing systems, and production testing. The requirements for this phase could, e.g., address

- hardware constraints
- interfaces to existing software or data base systems
- inference functionality, i.e., what queries the system should be able to answer
- response time

3.2.1 Use cases and user groups in the requirements engineering process

As discussed in Section 2, a use case driven approach to requirements engineering puts emphasis on the functional requirements of the system, by focusing on the interactions between actors (this being either persons or other hardware/software modules) and the system. This focus is consistent with the general objectives of the requirement

engineering process in AMIDST, and the use case-based approach to RE is therefore adopted in AMIDST. However, to obtain more well-defined requirements and establish a closer connection between the use cases and the project stages/work packages (c.f. Characteristics four), we further require that use cases should be specified as indivisible scenarios. Specifically, when defining the use cases, the industrial partners were informed that:

> *. . . a use case should ideally be indivisible. If a use case can be decomposed into multiple sub-use cases, each with a well-defined sub-objective relevant for AMIDST, then these sub-use cases should be described separately.*

The requirements derived from a use case are typically specified in relation to a particular user or type of user (possibly another component of the system). In AMIDST the possible users have very diverse backgrounds, ranging from developers with an intimate knowledge of the key technologies embedded in the AMIDST framework to programmers and users working in marketing. In order to ensure that focus is on the future *users* of the system, the AMIDST requirements engineering process also adopts and identifies user groups (as described in Section 2), which will be explicitly linked to the relevant requirements.

3.3 The general AMIDST requirements engineering process

To ensure a sufficient amount of knowledge transfer between the partners (c.f. Characteristics three, Page 5), the overall requirement engineering process will be carried out in an iterative fashion that is expected to involve a high level of cooperation and interaction between the partners.

In Figure 3, inspired by [14], an illustration of the requirement engineering process for AMIDST is given. The process contains five phases, which are discussed below.

Preparation I: This phase starts at the same time as work package one and ends when the initial template, see Appendix A, for the requirement engineering document is finished. In this template, the requirement engineering process is outlined, including definitions of use cases, user groups, and how to link requirements with the stages and WP/tasks in the development process. In order to meet characteristic three, four and five, the use case providers are asked to provide a detailed description of the system context that the AMIDST software is expected to operate in, identify user groups, describe use cases and requirements. In order to meet characteristic four, the requirements are linked to their associated work packages and tasks.

Elicitation: The distribution of the above mentioned template marks the initialization of this phase. Its aim is to get an initial high-level description of the different use cases and their requirements. This information are specified by the use case providers in collaboration with the academic partners, thus addressing characteristic one, three and four.

Once the use case providers return the present document with the requested information, feedback and informal meetings are expected to clarify and refine the information provided. At the end of the elicitation phase, the aim is to have a first coherent description of the requirements for each use case provider.

Prioritization: In this phase the use case providers complete and refine the document template used in the previous phase. This template explicitly links each of the requirements to the relevant work packages and tasks in the AMIDST project, thus providing an initial consistency check with the DoW framework (see Characteristic one). Moreover, the template allows the use case providers to give a fine grained prioritization of the relevant requirements for the AMIDST framework. Specifically, the use case providers are asked to rate each requirement in terms of whether it is a must, should, or could requirement:

Must (be) These requirements are expected by the use case providers and include properties guaranteed in the AMIDST DoW framework.

Should (performance) These requirements are expected by the use-case provider, but are not explicitly agreed upon.

Could (delighters) Optional requirements that will often be satisfying to have.

This high-level prioritization scheme is inspired by the Kano model correlating product development with customer satisfaction, see Figure 2. Within each of these categories, the use case providers should also make a more fine-grained prioritization by numerically weighting the different requirements on a scale from 0 to 100.

Validation: In this phase, the requirements from all use-case providers are collected to get the *big picture*. This includes a discussion among the members of the science project review group as to the extend in which the requirements can be accommodated and whether they collectively produce any potential conflicts, either internally or in relation to the DoW framework. Revisions and negotiations of the detailed requirements are therefore expected. In this phase, it is important to ensure that Characteristic one is met.

Evaluation and Testing: In this phase, the focus is on the elicitation of the evaluation and testing procedures in the AMIDST project. This phase starts with the distribution of a new document template, where the aim is to obtain a high level description of the evaluation and testing methods that are necessary to measure the performance of the AMIDST framework. This phase is not strictly part of the requirements engineering process, but will supplement the process by providing detailed specifications of how to perform specific tests and evaluations. Documentation of this phase is out of the scope of the present document, but will be included in the initial version of the AMIDST handbook (Deliverable D1.3).

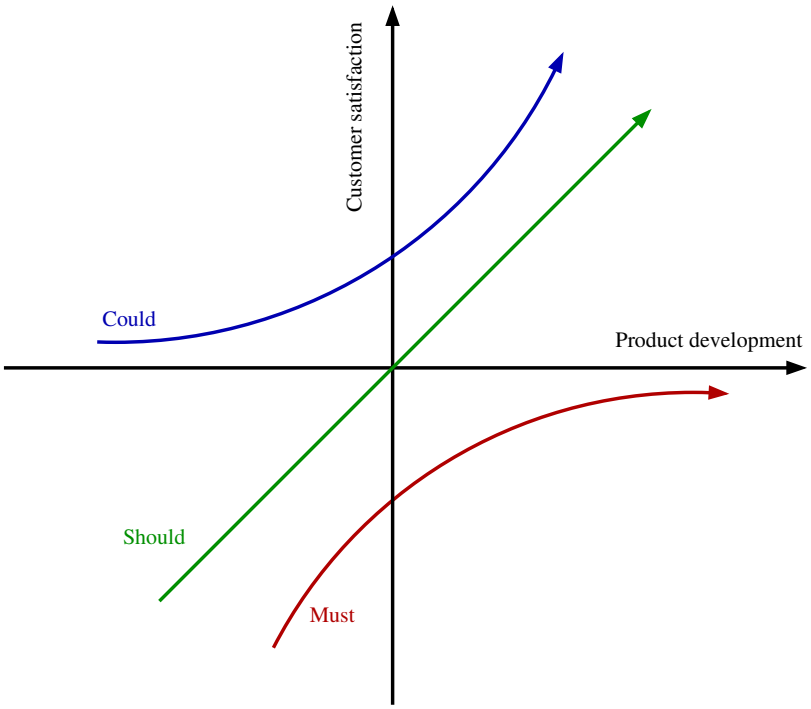

Figure 2: The Kano model.

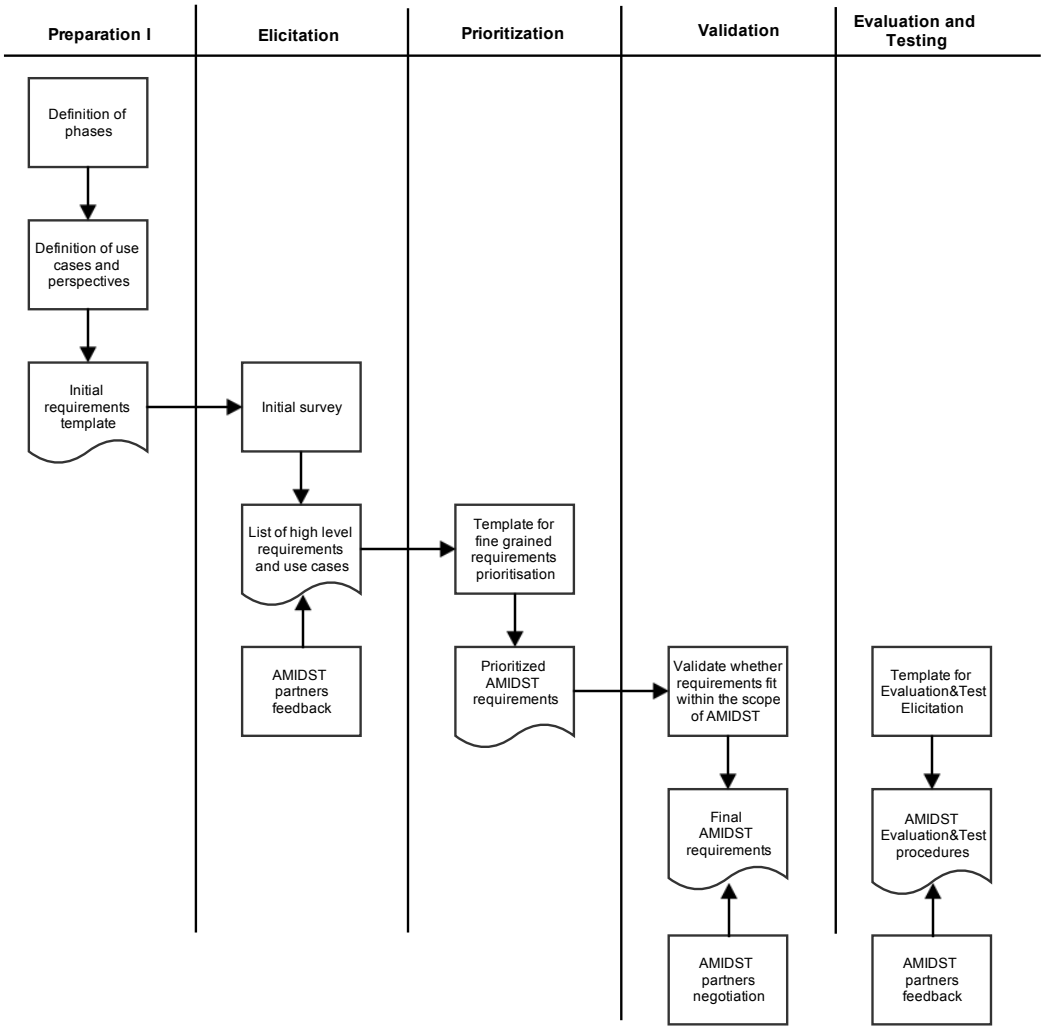

Figure 3: Description of the five phases in the requirement engineering process in AMIDST.

4 Realization of the requirements engineering process

In this section we describe the realization of the AMIDST requirements engineering process.

Division of work

For each industrial partner, a mentor among the academic partners is assigned. The mentor for Verdande Technology is NTNU, the mentor for DAIMLER is AAU; and the mentor for CajaMar is UAL. Hugin has a coordination role. We considered geographical and affinity reasons for these assignments. On one side, this allows each of the academic partners to focus on only one industrial domain. On the other side, the industrial partners have the academic support they need for identifying proper use cases. This division of work is a tool to mitigate characteristic two, because it eases the knowledge transfer between the industrial and academic partners.

Document template

Because of characteristics two, we have decided to not follow a RE process that heavily relies on personal meetings and direct personal interviews. Even though meetings still is an important ingredient in the process, we decided to distribute a document template to each of the industrial partners early in the process. The document template contains a description of how to describe the system context that the AMIDST software shall run in. Moreover, it contains a description of how to write the use cases, how to define the user groups and how to document the requirements. The main advantages with this template are:

- Industrial partners are pushed to contribute with use cases early, meaning that issues and misunderstandings are revealed early. This is important to mitigate characteristic three.
- All ideas are documented and there is no loss of information, which is common in interviews. This is important to meet characteristic one, three, four and five.
- The provided information can easily be transferred from the mentors to the other academic partners. This is important for meeting characteristic four.
- The partners can spend more time on use cases and requirements, before talking to the mentors. To some extent, this meets characteristics three, because it is easier for the mentors to learn when the requirements are more though through.
- The work on the different templates can be asynchronous in time. This eases the resources allocation for the different partners. This is related to characteristic two.

Requirements and work packages allocation

To make sure that all requirement comply with the document of work that is agreed upon upfront, every requirement is linked with the deliveries that are relevant. This is clearly a tool to mitigate characteristic one.

Requirements prioritization

In the Amidst RE process, the industrial partners are asked to categorize every requirement as an either must, should or could requirement. Moreover, they are also asked to give rigorous prioritization of each requirement. The intention is that the industrial partners are forced to think thoroughly about every requirement and how it relates to solving their problems. This is a way to mitigate characteristic five.

5 Conclusion, observations and reflections

Based on the lack of agreement in literature and in practice on a unified RE process for small projects, we chose to make our own process. The approach was to identify the main characteristics in the AMIDST project that is believed to have an impact on the choices for how to conduct the RE process. The main characteristics are that there is a pre-specified scope in the project, many different stakeholders, the software is supposed to solve very different problems and that the targets are not fully defined. A use-case driven approach was chosen and we also introduced a document template early in the project to mitigate the above challenges. The project is not finished yet, so it is difficult to fully evaluate the process today. However, it is positive that there seem to be a high level of agreement between the partners at this point.

References

[1] Quispe, A., Marques, M., Silvestre, L., Ochoa, S.F., Robbes, R.: Requirements engineering practices in very small software enterprises: A diagnostic study. In: Proceedings of the 2010 XXIX International Conference of the Chilean Computer Science Society. SCCC '10, Washington, DC, USA, IEEE Computer Society (2010) 81–87

[2] Aranda, J., Easterbrook, S., Wilson, G.: Requirements in the wild: How small companies do it. In: Proceedings of the 15th IEEE Requirements Engineering Conference. (2007) 39–48

[3] Nikula, U.: Phd thesis: Introducing basic systematic requirements engineering practices in small organisations with an easy to adopt method. Technical report, Lappeenranta University of technology (2004)

[4] Nikula, U.: A state-of-the-practice survey on requirements engineering in small- and medium sized enterprises. Technical report, TBRC Research Report (2000)

[5] Dorr, J., Adam, S., Eisenbarth, M., Ehresmann, M.: Implementing requirements engineering processes: Using cooperative self- assessment and improvement. IEEE Software **25**(3) (2008) 71–77

[6] Dingøyr, T., Dybå, T., Moe, N.B.: Agile Software Development: Current Research and Future Directions. Springer (2010)

[7] Kavitha, C.R., Thomas, S.M.: Requirement gathering for small projects using agile methods. IJCA Special Issue on Computational Science - New Dimensions & Perspectives (3) (2011) 122–128

[8] Pohl, K.: Requirement Engineering - Fundamentals, Principles and Techniques. Springer-Verlag (2010)

[9] Cockburn, A.: Writing Effective Use Cases. Addison-Wesley Professional (2001)

[10] : Ieee standard glossary of software engineering terminology. IEEE Std. 610.12-1990 (1990)

[11] Sommerville, I.: Software Engineering. Addison-Wesley (2011)

[12] Jacobson, I.: Object Oriented Software Engineering: A Use Case Driven Approach. Addison-Wesley (1992)

[13] Eigner, M.S.: Product Lifecycle Management. Springer-verlag (2009)

[14] Ebert, C.: Systematisches Requirement Engineering. Dpunkt Verlag (2010)