Requirements Engineering in Small and Medium Enterprises: State-of-the-Practice, Problems, Solutions, and Technology Transfer

Erik Kamsties[1], Klaus Hörmann[2], and Maud Schlich[2]

[1] Fraunhofer Institute for Experimental Software Engineering (IESE),
Sauerwiesen 6, D-67661 Kaiserslautern, Germany
kamsties@iese.fhg.de
[2] Software Technology Initiative (STI e.V.),
D-67661 Kaiserslautern, Germany

Abstract. Little is known about requirements engineering practices in small and medium enterprises (SMEs). This paper summarizes the results of a workshop on requirements engineering held with practitioners from 10 SMEs. The current state-of-the-practice, as reported by the practitioners, differs significantly, and so do their individual problems due to contextual issues (e.g., in-house vs. contract development, type of product, etc.). The participants were presented with a set of current requirements engineering principles, techniques, methods, and tools. Important concepts were motivated by small case studies and experiments which we experienced as a good vehicle for technology transfer. The design and the results are described as well as the practitioner's rating of the techniques and methods.

1 Introduction

A few studies report on requirements engineering practices in industry. The most cited study up to now was performed by Boehm in the early 80ies ("1-10-100 rule"[1]) [1]. Recently, some field studies provided detailed insight, like the study by Lubars et.al. [2] about RE in general and by Gerhart et.al. [3] about formal methods in particular. However, little is known about the requirements engineering practices specifically in small and medium enterprises.

Small and Medium Enterprises[2] (SMEs) are often considered "motors" of industrial growth: They are very dynamic, innovative and efficient and - from a Software Engineering consultant's point of view - very interesting but also demanding customers with high growth potential. For instance, some SMEs develop surprisingly large software systems of considerably complexity (e.g., one system

[1] Catching errors in the requirements phase costs 10 to 100 times less than catching errors in the development and maintenance phases.

[2] Usually a company is considered as an SME if it has less than 500 employees, less than 50 MECU annual turnover and not more than one third of the company is owned by a non-SME.

Published at Conference on European Industrial Requirements Engineering (CEIRE'98), 19-20th October 1998, London, UK

we are aware of has 2,000,000 lines of code). As compared to large companies, however, SMEs have very specific problems:

- The maturity level in Software Engineering is very low.
- Employees and management are typically overwhelmed with day to day business, leaving little space for strategic issues such as quality and process improvement.
- There is a large demand for know-how transfer with respect to basic issues and "how to do it".
- Owners and management are often not used to co-operate with external consultants.

However, the rising number of tools for requirements management and modeling on the market can be taken as an indication that SMEs are just now moving from ad hoc software development to more systematic practices (this conclusion is based on the assumption that big companies do already have a stable requirements engineering process).

The Software Technology Initiative (STI e.V.) is an association of more than thirty companies, consultants, and public organizations which offers consulting and a variety of seminars, training programs, and workshops mainly for SMEs within the software and embedded systems industry. This study is based on an STI workshop on requirements engineering. Our general experience with SMEs within this framework is that SMEs

- do not subscribe to large co-operation projects, but rather to small slices of work
- require well-packaged and mature results being transferred in short time periods
- are rather price sensitive.

The STI workshop on requirements engineering consisted of five sessions, with each session lasting four hours. Each session consisted of an experience report from one of the participants, a lesson on a requirements engineering topic given by us, a small case study or experiment according to the topic, and discussions. We tried to keep the lessons as interactively and lively as possible.

In this paper, first, we sketch the state-of-the-practice in the accidental sample of SMEs which were participating in our STI workshop on Requirements Engineering. We give an overview on the topics that were presented in the workshop and the rationale for selecting them. The novel aspect of the workshop is small case studies and experiments with the practitioners to motivate requirements engineering improvements. We outline these studies and summarize the feedback of the practitioners. The paper ends with a summary and conclusions.

2 State-of-the-Practice and Problems

This chapter describes the state-of-the-practice we found in the accidental sample of SMEs in our workshop. The information provided below is based on self-

presentations given by representatives of the companies and discussions that evolved at the workshop.

The SMEs participating in our workshop are operating in a wide range of application domains: business information systems (4 SMEs), geographic information systems (2), multi-media systems (1), avionic systems (1), software for clothes manufacturing machines (1), and mathematical software for civil engineers (1). The participants themselves were project or group managers, requirements engineers, and software developers.

The subsequent discussion of the state-of-the-practice is organized around the following issues: product and project characteristics, requirements sources and elicitation activities, requirements engineering products (i.e., the software requirements specification (SRS)), requirements verification and validation activities, and system evolution (i.e., further development activities).

2.1 Product and Project Characteristics

The requirements engineering practices we found differ a lot between the SMEs. Among the major influencing factors are:

1. Type of system: market-driven vs. customer-specific (bespoken) systems,
2. degree of adaptability: user-configurable vs. vendor-configured systems, and
3. type of software development: in-house vs. subcontracting.

The majority of the participating SMEs (9 out of 10) develop market-driven systems; one company develops customer-specific systems for in-house application. Two different product development strategies for market-driven systems were visible: (1) starting directly with a market-driven system and (2) generalizing a solution for one initial customer to the needs of other similar customers. SMEs that chose the second strategy were in different stages of that generalization process. Some of them were in the initial stage, developing a single system with the perspective to generalize it later, others already had a generalized solution.

Each of the SMEs develops software products that must be adaptable to the customers' and users' needs. The degree of adaptability ranges from user-configurable issues as the appearance of the graphical user interface to vendor-configured issues as specific functionality required by individual customers. The size and complexity of the software product developed is considerably high, one SME mentioned that their software has 2,000,000 lines of code, others reported similar size. Also the application logic (e.g., information architecture, state space, or possible feature interactions) are quite complex in business as well as in technical application domains.

Projects in SMEs are usually triggered by (1) technology changes (e.g., new operating/window system, new hardware), (2) new features to be integrated due to market pressure, or (3) customer-specific adaptations. We did not hear about any software developments from scratch.

A surprise for us was the fact that two SMEs are experienced in developing software systems with subcontractors in Europe and also the U.S..

2.2 Requirements Sources and Elicitation Activities

Requirements sources are customers and users, observation of the market, and in-house domain experts. The degree of customer/user involvement depends on the type of the system (i.e., market-driven vs. customer-specific system). Except for the one particular SME developing customer-specific software, no SME reports intensive customer or user involvement.

The product development strategy for market-driven systems (see subsection 2.1) has an impact on the requirements elicitation process. The requirements are typically self-invented in the case that a company starts with a market-driven product. In the initial phase of the generalization process, interviews with the prototypical customer are conducted and domain experts are involved. Workshops are held with potential and existing customers to elicit further requirements when the product is to be generalized. The ultimate goal of these workshops is to reach as much consensus as possible between all customers about the desired features. An additional benefit of workshops can be that potential customers get deep practical insight into the software product that they are going to buy. We did not hear of any severe problems with requirements elicitation.

2.3 Requirements Specification, Verification and Validation

The documentation of the requirements in a software requirements specification (SRS) strongly depends on the type of software development (i.e., in-house vs. subcontracting) and the degree of adaptability (user-configurable vs. vendor-configured systems). The two subcontracting SMEs have, of course, an SRS. SMEs that develop vendor-configured systems usually state only the requirements of each particular customer-specific adaptation, because these become part of a legal contract with the customer. The core system has often no SRS because it was built in the 'dynamic' start-up phase of the SME.

There was much debate about the degree to which the requirements should be stated in the case of market-driven systems that are developed in-house (i.e., in the case where the requirements sources as well as the software developers are located in-house), because then the requirements can be communicated verbally. As a consequence, there is often no requirements document at all, but only a collection of meeting minutes and later on a user manual. Some SMEs try to use the user manual as an SRS, but encounter problems, because it is not detailed enough and exceptions are usually not stated in a user manual.

The SMEs that do produce an SRS are faced with more subtle problems:

- It is not clear how to specify a graphical user interface (GUI). Control flow representations are often useless because of the high number of possible control flows.
- The complexity of requirements documents makes them difficult to understand and to review. The complexity is even harder to grasp if the requirements document is written with a text editor and filtering and sorting of requirements are not supported adequately (as it would be with a requirements management tool).

We found modeling techniques (basically object-oriented approaches) in only two SMEs, but applied with completely different goals. The first SME uses models to find defects in their textual SRS, but the textual SRS remains the major output of the requirements phase. The second SME, to a large extent, captures the requirements in models, which are the major input for the development activities. The first SME simulates their requirements models and develops test cases in order to further increase the quality of the textual SRS. Besides this company, we did not hear of any systematic inspection, simulation, or verification activities at the requirements stage.

2.4 System Evolution

SMEs that do not produce an SRS did not claim any problems during implementation of the software, either. It seems that their software developers like 'creative freedom'. The trouble begins with testing, because the first task of the test personnel is now to elicit the requirements again in order to create test cases.[3] Thus, large effort must be spent on testing and also on rework, because conceptual problems are detected during testing that should appear during the requirements phase. The conceptual problems are sometimes hard to resolve. With respect to the maintenance phase, one problem was reported: The implementation of new requirements can cause unforeseeable interactions with requirements that are already implemented (an SRS would help finding such interactions before the new requirement is implemented and tested). Other problems due to this phase were not reported, which might be due to the fact that the employee fluctuation rate is not as high as in large companies. Adding new people to a project again is difficult. Because of the lack of documentation they must be coached intensively by experienced people. One participant claimed that it takes one year to get a new software engineer on board, plus the coaching effort for experienced team members.

Again, the SMEs that do produce an SRS are faced with more subtle problems. For instance:

- The domain knowledge implicitly contained in requirements makes the SRS difficult to understand for software developers. Furthermore, developers claim that textual requirements documents are not precise enough and often incomplete. This problem becomes bigger if the software is developed off site and, thus, defects are not detected before acceptance testing, since developers tend to 'interpret' open issues.
- Requirements are too vague or prosaic to be testable. Therefore, the requirements engineers must also work as testers, since they know how the system should behave.
- Requirements are not traceable (i.e., a change to a requirement would affect virtually all components of the system).

[3] Testing market-driven systems is difficult, because the users are not well-known, and thus, the usage of the software is not well-understood. Therefore, often the software cannot be tested thoroughly.

As mentioned before, most of the SMEs develop adaptable software products. However, none of the SMEs have a strategy to deal with adaptability explicitly (e.g., a product line approach). Those SMEs, which developed and improved their products over several years, found pragmatic ways to ensure adaptability. Some SMEs solved the issue on the architecture level by maintaining a core system and implementing customer-specific adaptations as additional modules. Others solved the issue at implementation level with configuration files that are read by the software at runtime. The individual way depends on the application domain and the required degree of adaptability.

3 Requirements Engineering Process Improvement

Driving factors in SMEs for requirements engineering process improvements are the problems with testing and ISO 9001 certification. Main road blocks for technology transfer are the small budgets and tight project schedules. Once the software product matured on the market, the pressure to improve requirements engineering activities for this product is not so high anymore.

The basic goal of the workshop was to cover the whole requirements engineering life-cycle beginning with elicitation, proceeding with documentation, and ending up with validation/verification, and to give an impression of the whole repertoire of proven requirements engineering principles/guidelines, techniques, methods, and tools. From this goal, we derived six topics for the workshop, which are summarized in Table 1.

4 Case Studies and Experiments

All topics of the workshop were motivated with small case studies and experiments (except 'legal aspects of tendering/subcontracting'), with one case study being performed per session. One experiment was performed, which took two sessions.

4.1 Case Study "Classifying and Checking Requirements"

The goal was to motivate a well-structured SRS and to provide examples of 'good' and 'bad' requirements. Each participant received an unordered, unstructured statement of requirements, an outline of an SRS, and a form to record defects in requirements. The task of the participants was to classify requirements according to typical categories (functional requirements, non-functional requirement, domain property, etc.), to assign them to the appropriate section of the SRS, and to check the requirements for the usual qualities.

The result of the case study was that the value of a well-structured SRS compared to an unstructured document was recognized. However, less agreement could be reached about the question in which section to put a requirement. Only a few defects in the requirements were found ad hoc.

Table 1. Outline of the Workshop

RE Topic	Details
SRS Improvement	Motivation for an SRS
	Economic aspects (cost savings for rework activities after testing)
	Contents (glossary, etc.)
	Structure (e.g., IEEE-Std. 830-1993)
	Qualities (correctness, completeness, consistency, clarity, traceability, etc.)
Elicitation	Identification of requirements sources
	Elicitation techniques (interviews, workshops, surveys, etc.)
	Planning and conducting interviews
	Use cases for user-oriented documentation of requirements
Inspections	Inspection process (planning, preparation/reading, meeting, rework, follow up)
	Reading techniques (perspective-based reading [4])
Modeling Requirements	Motivation and benefits
	Models and pragmatic notations (environment models/ context diagrams, architectural models/ block diagrams, behavioral models/ sequence charts, etc.)
	Overview on methods (OO modeling and specialized methods, e.g., Statecharts)
Requirements management with tools	Principles of requirements management (baselines, traceability, etc.)
	Tool overview
	Introduction to a particular requirements management tool
Legal aspects of tendering and subcontracting	Principles
	Types and elements of contracts (German Law)
	Problems with fixed price contracts while the SRS still needs to take shape

4.2 Case Study "Requirements Elicitation and Elaboration"

The goal of this case study was to motivate systematic methods to elicit and elaborate the requirements of a system. The participants were divided into 'customers' and 'requirements engineers'. The customers received a full description of a system, the requirements engineers received only a sketchy description. The task of the requirements engineer was to prepare and conduct an interview with the customer in order to elaborate the requirements (based on the "5W1H-analysis", i.e., asking why, what, when, where, who, and how questions). The task of the customer was to answer the questions.

The result was that most participants agreed on systematic methods for requirements elaboration. They were not able to elaborate most of the requirements of the system within the given time frame, because they rushed into details instead of developing a complete (high-level) view of the system. We used this case study to motivate use cases, but it remains an open question whether use cases could help to a large extent, since domain knowledge plays an important role in requirements elaboration, too.

4.3 Experiment "Requirements Validation"

The goal of this experiment was to compare different techniques for finding defects in requirements documents, namely inspections (with checklists), and requirements modeling. In the first session, the participants received a textual, well-structured requirements document, a checklist, and a defect form. The task was to apply the checklist. In the second session, the task of the participants was to create a set of models from the previously mentioned SRS. Again, they had to record all defects on a defect form.

The participants found a lot of defects with checklists, but mainly clerical or obvious ones. The essential defects remained uncovered. Interesting for us was the fact that most of the defects were mentioned only by one participant each, which confirms that it is difficult to tell what a defect in a requirements document is, as opposed to code. The creation of requirements models forced the participants to analyze the requirements in depth and, hence, they found more essential defects than with checklists. But, unusual to practice, the participants read the SRS twice, which might be another reason why they found more essential defects the second time.

4.4 Case Study "Tool Support"

Here, the goal was to explore the value of requirements traceability, and the potential of a database to manage requirements and the links between them. The participants received a short training with a commercial requirements management tool. The task was to analyze the impact of several requirements as well as to trace implementation information back to the requirements stage. Other tasks included the use of views to extract information needed for particular tasks, access control, and versioning/baselining of requirements.

Traceability was considered worthwhile, but the effort to introduce and maintain it was discussed controversely. The application of requirements management tools was considered quite useful.

5 Rating of RE Topics

At the end of the workshop we asked the participants to rate the content according to two questions: (1) Which topics are relevant/not relevant to your environment in general? (2) Which topic has the highest priority within your improvement plan? Table 2 below summarizes the results. The numbers in the second, third, and fourth column indicate how many *participants* (not SMEs) found a topic relevant or gave a topic the highest priority.

Table 2. Rating of RE Topics

Topic	Relevant	Not relevant	Highest priority
Improvement of SRS	10	2	2
Elicitation	7	5	1
Inspections	9	3	2
Modeling	*12*	*0*	*5*
Tools	7	5	2
Legal aspects	4	8	0

The following ranking of topics with respect to 'relevance' and 'priority' can be derived from Table 2:

- Relevance: Modeling > Improvement of SRS > Inspections > Tools = Elicitation > Legal aspects
- Priority: Modeling $\gg$ Improvement of SRS = Inspections = Tools > Elicitation > Legal aspects

'Legal aspects' and 'elicitation' are ranked as least relevant, which is probably due to the fact that most SMEs develop market-driven systems in-house. Those who let subcontractors develop their software seem to have no severe problems with legal aspects. Tool support was seen as quite useful, but not as most pressuring, which might be related to the fact that most of the SMEs do not have an extensive SRS. 'Modeling' is the most relevant topic that most of the participants want to improve. Satisfying for us was the fact that most of the SMEs see the importance of introducing an (or improving the) SRS, which is the basic requisite for all other activities in requirements engineering.

6 Summary and Conclusions

In this paper we have summarized the state-of-the practice and problems of Small and Medium Enterprises (SMEs) with respect to requirements engineering, as

observed in a workshop with practitioners from 10 SMEs. We outlined several possible improvements and their assessment by the practitioners.

Most SMEs develop market-driven systems, which is a relatively new area in requirements engineering research. Most SMEs do not have an extensive SRS. They develop their systems in-house, and thus, are being able to communicate the requirements verbally. In turn, they spend a lot of time on testing, because in order to create test cases the first task of the test personnel is often to elicit the requirements again (since only a few requirements are documented). The amount of rework is also higher, because conceptual problems are detected during testing, which could otherwise be detected and removed during requirements analysis with much less effort. Furthermore, the lack of documentation makes it harder to add new people to a project. All workshop participants agreed on the importance of an SRS, but one major question arose: how detailed should an SRS be, if the software is developed in-house? Instead of describing requirements in detail textually, the SMEs are highly interested in modeling requirements, since models can be reused (to some extent) for design (as promised sometimes by object-oriented technology). However, which modeling approach shall be chosen for a particular project, and how this approach can be applied most cost-efficiently, are open questions. How to specify graphical user interfaces in a pragmatic way is an open question, too.

SMEs wish to produce an SRS with low additional effort. Since a user manual must be produced anyway, it may be worthwhile to study the relationship between the SRS and the user manual. If the SRS is a superset of the user manual, then the SRS could be organized in such a way that the user manual can be generated from the SRS automatically. Only low additional effort would be required to fill the sections of the SRS which are not contained in the user manual.

The software product of an SME was frequently developed in the company's dynamic start-up phase, thus, there is often no documentation about the core system, but only about recent adaptations. Therefore, another question is how to re-engineer a requirements document for an existing system.

We experienced SMEs as an area quite interesting for requirements engineering research, since the size and complexity of systems developed is comparable to those systems developed by large companies. However, tighter budgets require small improvement steps. Since most projects in SMEs are aimed at customization and integration of new features desired by the market, RE process improvements should be incremental and evolutionary instead of revolutionary. We found the experimental approach to requirements engineering as proposed in [5] useful for motivating practitioners for requirements engineering process improvements, and for enabling them to evaluate the applicability for themselves (according to the ideas of the Personal Software Process by Humphrey [6]). Demonstrated benefits of RE technologies are an essential step towards sustaining process improvements in a company's environment.

Acknowledgment

We thank Rupert Wiebel from Quality Systems and Software (QSS) Germany for providing us an evaluation license of DOORS for the workshop. Furthermore, we thank our colleagues at the Fraunhofer Institute, especially Peter Knauber, and Sonnhild Namingha for their comments on previous versions of this paper.

References

1. Barry W. Boehm. *Software Engineering Economics*. Advances in Computing Science and Technology. Prentice Hall, 1981.
2. Mitch Lubars, Colin Potts, and Charles Richter. A review of the state of practice in requirements modelling. In *Proceedings of the IEEE International Symposium on Requirements Engineering (RE93)*, pages 2–14, San Diego, California, USA, January 1993.
3. Susan Gerhart, Dan Craigen, and Ted Ralston. Experience with formal methods in critical systems. *IEEE Software*, pages 21–39, January 1994.
4. Victor R. Basili, Scott Green, Oliver Laitenberger, Filippo Lanubile, Forrest Shull, Sivert Sorumgard, and Marvin V. Zelkowitz. The empirical investigation of perspective-based reading. *Journal of Empirical Software Engineering*, 1(2):133–164, 1996.
5. Erik Kamsties and H. Dieter Rombach. A framework for evaluating system and software requirements specification approaches. In Manfred Broy and Bernhard Rumpe, editors, *Proceedings of the RTSE'97 - Workshop on Requirements Targeting Software and Systems Engineering*, pages 281–296, Munich, Germany, April 1998. Technical University of Munich (TUM), Technical Report TUM-I9807.
6. Watts H. Humphrey. *A Discipline for Software Engineering*. Addison-Wesley, 1995.