Agile Requirements Engineering for a Social Insurance for Occupational Risks Organization: A Case Study

Mario Pichler
SCCH - Software Competence Center Hagenberg
Hauptstrasse 99, A-4232 Hagenberg, Austria
mario.pichler@scch.at

Hildegard Rumetshofer
FAW Software Engineering gGmbH
Hauptstrasse 119, A-4232 Hagenberg, Austria
hrumetshofer@faw.uni-linz.ac.at

Wilhelm Wahler
AUVA - Social Insurance for Occupational Risks
Adalbert-Stifter-Strasse 65, A-1201 Vienna, Austria
wilhelm.wahler@auva.at

Abstract

The objective of this paper is to report on the challenges and experiences gained during a three years multidisciplinary software development project in the insurance domain, focusing on the employed requirements process. Although, there has been consensus on an agile development process among the project partners, this approach stands in some contrast to the traditional (standardized or regulated) development processes commonly employed in the insurance domain, or general, in the public sector. The lessons learned of applying an agile requirements process under the conditions of traditional processes at the customer's site; geographically distributed offices of the customer and development team; diverse interests of involved customers' departments, administrative and operational staff; limited availability of field workers etc. are provided as recommendations for other research and development institutions. Thereby, the viewpoint of each project partner, including the customer's perspective, is described.

1 Introduction

IT systems development in the public sector can commonly be characterized as highly-regulated [7] and based on standardized lifecycle process models. The usage of the V-Model [6], the development standard for IT systems of the Federal Republic of Germany, is quite common for IT systems development in the public sector in Germany, and is also used in an adapted version by the public administration in Austria [3]. However, recent advances in software development suggest to use more agile approaches [1, 4], than is the case with the usage of heavy-weight process models as mentioned above.

In a project with the nation-wide operating Austrian Social Insurance for Occupational Risks (AUVA) [2]—the customer—, the AUVA and two research and development institutions agreed to apply an iterative, evolutionary process for the development of an enterprise-wide software system.

In this paper, the challenges and experiences gained when applying an agile software development approach in a domain where traditional (standardized or regulated) development processes are common, are given. As the project has been initiated in summer 2003, a considerable collection of experiences can be discussed so far.

Two papers of recent occurrences of the IEEE RE conference series are taken as discussion base for the paper at hand. The first is *"What influences the Requirements Process in Industry? A Report on Industrial Practice"* by Alexander et al. [7]. We add our experience, by also going beyond the experiences of the AUVA project. The second paper is *"Understanding Requirements in Enterprise Systems Projects"* by Gulla [12]. In the latter paper, a number of challenges of enterprise systems, we are also faced with from a requirements process point of view, are described. Gulla's paper is also interesting for us, when asking the question what it means to understand enterprise systems requirements when using an agile requirements engineering process.

Following, in Section 2, we describe the field of activity of the Austrian Social Insurance for Occupational Risks and state the project goal. In Section 3, the (agile) software engineering process negotiated at the very beginning of the project among the partners, is presented. Following, in

14th IEEE International Requirements Engineering Conference (RE'06)
0-7695-2555-5/06 $20.00 © 2006 IEEE

Section 4, the experienced requirements engineering challenges by applying this approach are described. The lessons learned when dealing with these challenges, are given in Section 5. A summary of challenges and associated recommendations concludes the paper.

2 Domain and Project Goal

The AUVA is the Austrian Social Insurance for Occupational Risks for approximately 3 million employed persons and 1.3 million school children and students. They care for about 180.000 victims of occupational accidents and diseases annually. About 300.000 persons involved in accidents of all kinds are treated in their trauma centers and rehabilitation centers every year. They pay about 73.000 compensations per year to victims of occupational accidents and diseases or their families. Therefore, a central motivation of the AUVA is the prevention of occupational accidents and diseases.

One major element of the prevention activities are the visits of prevention workers (experts in a specific domain, e.g. noise control, and medicine) in companies, to suggest them possibilities to improve the employees' safety and health conditions at their workplaces. In the regular case, visits of prevention workers are demanded by a company, e.g. when a new manufacturing system has been installed and the company (and/or the company's works council) wants measurements of the resulting, new noise level in the construction hall. Moreover, the AUVA is bound by law to serve companies all over the country in this regard. To be able to fulfill this responsibility, the AUVA with its headquarters (HUB) in Vienna, has four further offices (LS: WUV, LUV, GUV, SUV), each responsible for one or more provinces. In case that an LS is responsible for more than one province, there are further subsidiaries in the respective provinces.

2.1 Project Goal

To fulfill the abovementioned obligation, and, beyond it, to further increase the number of served companies, AUVA seeks to improve its business processes for faster order processing etc. An internal goal of one department—the noise control department—is to visit all served companies in a regular interval of five years. As the number of prevention field workers is limited, it is challenging for this department to stick to this five year's period. To guarantee the satisfaction of served companies, also other AUVA departments need to improve their processes.

Following, the goal of the AUVA for the project described herein, is to provide software support for—up to now, partly manually handled—work processes in the 'prevention' domain. The core prevention processes, which are slightly different in the single offices are unified on a *best practices act as template for a standardized process* basis. This means that there have been drafts for process unification suggested by the headquarters, however, well-established processes in different offices were also welcome as basis for improvements.

Another requirement has been the adoption of state-of-the-art software development processes within AUVA, as its own IT staff members are experts on various systems already in use, however, they are not sufficiently familiar with recent methods for the development of an enterprise system as required. Furthermore, usage of latest technologies (as well as replacement of obsolete ones) is required. Aiming at 1) developing sophisticated tools to support prevention (administration and field) workers' daily tasks and 2) a know-how and technology transfer into the AUVA, collaboration with two research and development institutions (contractors) has been signed.

At this point, we come back to Alexander et al. [7], where one of the survey results was that collaborations with research organizations have little effect on industrial requirements practice in general. We can observe here that also in a sector with regulations and standards, collaborations with research organizations seem to be fruitful for organizations. In our special case, the contractors moderated nearly all of the meetings among the different departments of the customer with the aim to unify different business processes—a somewhat critical endeavor.

3 Negotiated Innovative SE Process

A major requirement of the AUVA has been that the developed software represents a significant assistance for the users, rather than to require additional workload from them. A simplification of daily tasks for prevention workers has been communicated as an overall goal. Based on this requirement, after an evaluation of the suitability of available enterprise systems software, also with a focus on customization possibilities [12], it was decided to newly develop a system satisfying the special needs of the customer. Especially, a user-centered approach with ongoing interaction and short feedback cycles to align the supporting system on the stakeholders' needs has been identified as paramount.

Rather than following a highly-regulated and, most notably, a rather heavy-weighted course of action, as has been observed for projects in the public sector applying the V-Model [3, 6], the responsible project stakeholders agreed to utilize a more flexible approach. A software development process based on best practices of the Microsoft Solutions Framework (MSF) [5] has been negotiated among the stakeholders in the very beginning of the project. As important information, we want to mention that this approach has been negotiated among the contractors and the IT department of

the customer company. We come back to this important point in the following sections.

In more detail, the approach we followed was driven by short iterations encompassing process steps such as analyzing the customer's vision, designing visual models to extract requirements for preparing stop-or-go-decisions as well as architectural design, development and test of implemented features. Each iteration was started with an iteration planning meeting and ended up with evaluation workshops and feedback given by the customer/users. This iterative, evolutionary, user-centered development process, allowing strong user participation and short feedback cycles has been considered as optimal for reaching the project goal.

It is not our objective to discuss the differences of formal versus agile software development processes in this paper. However, the usage of agile methods implicates a number of challenges, especially for the requirements process. One major difference is that no detailed requirements specification (product and functional specification) which had to be signed by the customer and contractors as basis for further development was created in an early, long-lasting analysis phase. The resulting challenges, we were faced with using an agile approach in this project, are explained in the following section.

4 Challenges for RE Process

The challenges we have been faced with during the past three years of developing an enterprise-wide software system supporting the business processes of the customer, are described briefly in the following.

Challenge (C1). *Traditional processes at the customer's site.* Using an agile development approach might be well-understood by software developers. However, such an approach is mostly unknown in other industries like machine building, building industry or road making and further domains, where more formal processes are established—in our case, in the insurance domain.

Challenge (C2). *Misunderstandings of prototypes and final product.* Closely related to (C1) is the misunderstanding of software prototypes and the final product version. Other than a car prototype, where almost everyone assumes that the basic functionality is already built in and beautification and comfort are still to implement, it is different in software development. Especially, the meaning of user interface prototypes has to be explained clearly.

Challenge (C3). *Balancing management, workflow, and technology tier.* According to [12], the requirements must take into account the needs of the management, the process organization, and the technical staff. This is especially the case, if priorities are changing during the course of the project—a principle that an agile requirements process should be prepared for.

Challenge (C4). *Geographically distributed offices of the customer.* This fact makes it harder to establish the requirements engineer as single intermediate between customer and development team and to organize user tests etc.

Challenge (C5). *Diverse interests of involved customers' departments, administrative and operational staff.* Besides divergent desires and needs of stakeholders, one can also observe different levels of stakeholder motivation. Some have very positive expectations. Other ones have more negative attitudes towards the new system.

Challenge (C6). *Limited availability of field workers.* Onsite customer is a basic principle of agile software development. The advantage is that both customer and development team work together and questions of the development team (primarily concerning customer's requirements) can instantaneously be answered by the customer. However, it is simply not possible for field workers to be continuously available for questions.

Challenge (C7). *Distributed development team.* Developing a complex software system, as is the case with enterprise-wide software, involves a number of partners working together to fulfill all stakeholders' desires and needs. Similar to (C4), a distributed development team represents a challenge not only for the requirements engineer.

5 Lessons Learned

By facing these challenges as described in the previous section, we captured a set of experiences, which we want to share within the RE community. We believe that these experiences can be advantageous for other research groups and development institutions applying an agile requirements process, especially, under the conditions of traditional processes at the customer's site. Consequently, we provide ten recommendations for consideration when conducting an agile development process for your software projects.

Recommendation (R1). *Demonstrate objectives and appropriateness of agile development processes clearly to your customer.* Agile approaches are quite new in software engineering. Furthermore, they are in contrast to traditional (standardized or regulated) development processes commonly employed in the insurance domain. They are even totally unknown in traditional industries e.g. machine building, building industry or road making. Hence, a big challenge is to create awareness and trust for the benefits of agile approaches within these groups—sometimes a recurring task. We recognized that an agile approach is difficult to understand in case that the management changes at the customer's site. Such changes always request for explanations to create trust. Consequently, we suggest to clearly

defining the partners' goals when conducting an agile development process. As there are no detailed contract documents (e.g. detailed specifications), which could be shown to a new manager (as in our case), the partners should instead sign a contract defining commitment and objectives for an agile approach in an early stage. This document can serve as helpful artifact to get an idea of agile development for new members steering or joining the team.

Recommendation (R2). *Clearly describe the difference of (development) prototypes and the final product to your customer. "People do not really know what they want until they try it"* [13]. According to this statement, prototypes offer a pretty good possibility to identify stakeholders' desires and needs. Stakeholders pretty well understand the meaning of use cases and use case models—discussing them helped us to identify a huge number of requirements in a very short time frame. Anyhow, a superior way to collect requirements and evaluate the right understanding of requirements by the development team are prototypes, with which future users are able to experiment with. However, before user tests, you have to state clear that the prototype does not provide the whole functionality. Otherwise, users expect a running system, especially, in later project phases.

Recommendation (R3). *Make sure to establish the requirements engineer as single intermediate between customer and development team.* If you are practicing "too many cooks spoil the broth" in the context of requirements engineering, you will soon identify that your development team is in troubles. Grasping information from different persons (having different roles) at the customer's site by different persons (also having different roles) at the other partner's site complicates or even prevents the successful bundling of requirements. These multiple information channels eliminate well-regulated change management processes. This will be especially serious if flexibility for changes is paramount—as this is the case in agile approaches. Besides, it is evident that misunderstandings are pre-assigned when users and developers talk about requirements. They speak varying languages, having different judgments and ratings. The bottom line is that requirements are misinterpreted, wrongly implemented and finally revised. Time, money and efforts spent in vain. Consequently, it is important to guarantee the requirements engineer's attendance in meetings with the customer and to constitute the RE team supporting the requirements engineer as not too small.

Recommendation (R4). *Be aware of an adequate and concise management of your requirements information.* The concentration on the requirements engineer as single intermediate allows storage of grasped information into a single pool and to keep it up-to-date. A central access point to project information and development artifacts simplifies or primarily allows defining baselines and traceability of requirements for a particular iteration. If you are (in any case the customer is) interested to track, for instance, which requirement was realized in which iteration, single sourcing will serve as powerful instrument. This goes along with the existence of a single tool to support the RE process. To cache requirements information in different sources such as project Web page, project folder, RE tool, versioning system or developers' heads is both inefficient and a perfect source for misunderstandings. Given 73 meetings with customers, users and other stakeholders up to now, we realized the necessity and importance of concise requirements management to ensure no information losses.

Recommendation (R5). *Avoid a lengthy specification phase in early project phases.* Having a sketchy vision at the beginning of the project may guide you to too detailed analysis for a particular subarea in this early phase. This is especially the case when you are working closely with user groups not representing whole customer's site—changes are predetermined when other stakeholders are called in. Suddenly, high-priority requirements get unimportant. In the meanwhile, requirements are changed. When concentrating too long on eliciting information for a specific feature, there may be too less time to consider validated requirements for other features if different goals at the customer's site get significant. It is evident that this absence negatively effects further project phases. Therefore, a more agile approach with detailed analysis, when actually needed, is both convenient and purposeful, especially, with regard to projects having loose visions at their beginnings and changing priorities at run time.

Recommendation (R6). *Pay tribute to requirements modeling as quality assurance technique.* Unfortunately, software project managers often stress testing of the running application as main and single quality assurance technique. Consequently, writing automated tests for already implemented features completes the developer's daily job. However, these are functional tests, but what about logical ones? Getting an idea of the users' goals and the system's behaviors in an early phase—even on a high-level—is as important for the user as for the developer. The construction of usage model, initial domain model or user interface model as soon as possible in the development process allows to identify very early specification inadequacy and bottlenecks as well as nonconformity of interaction mechanisms [8, 11]. As an agile approach assumes the existence of short iteration cycles and the competency for grasping new or changing information quickly, requirements modeling will help to make a big step forward to accurate system quality.

Recommendation (R7). *Don't keep your focus simply on coding.* The choice for an agile development process assumes that requirements are not at once determinable and iterations are executed in cycles where each of those deals

with analysis, modeling, design, implementation and evaluation. Consequently, all these phases within a cycle request for manpower—an invitation for building up large teams already in an early phase. However, project teams encompassing too many developers in early project phases allure to concentrate on coding too soon [10], because project managers aspire to keep developers busy. Indeed, missing requirements, modeling and design avoid meetings among requirements engineer, designers and developers. Hence, developers will code on assumptions and not requirements. Surprises will be predetermined if changes are requested by the customer. Having visual models for usage, architecture and features prevents from starting with coding too early simply because manpower is bound to these tasks. Furthermore, it facilitates adjustment and understanding of new team members.

Recommendation (R8). *Treat users as your partners and keep them integrated in your project.* As mentioned above, the AUVA represents a variety of different user groups, interests and daily working processes. These organizationally and geographically distributed user groups are challenging for organizing user tests. However, meetings where users test (paper or running) prototypes are goldmines for identifying new and changing requirements [12]. Primarily, users will easy comprehend system's behaviors if they are allowed to play with something. Users' participatory development of paper prototypes improves acceptance and understanding for the final system because they track continuously step-wise progress, failings and improvements. User involvement allows fast and cheap reactions to changes. However, these goldmines can be tricky. Especially, if the customer's management who is responsible for determining and prioritizing favored features and requirements does not participate in these meetings (or even arrange completely different requirements with the project management), the development team will be stuck between a rock and a hard place. What should be done: realize users' desires, customers' demands or project managers' tasks? With regard to the ambition of following a user-centered approach, a clear understanding about who represents a particular user group and who has the power of decision is evident for all participants (users, stakeholders, managers). We realized that a single user representative equipped with the power of decision for each varying user group interacting close with the requirements engineer will be beneficial for passing the system's acceptance test.

Recommendation (R9). *Identify required development artifacts very early and consider the user's manual as requirements specification.* An agile approach requests for well-advised consideration and portioning of required development artifacts such as contract document, modeling diagrams, user's manual, coding or documentation. Hence, the selection for those artifacts is already crucial at the beginning of the project. Each selected artifact has to be kept alive, otherwise, they are useless and wastage. Each of them requests for manpower. We noticed that the concentration of artifacts depends heavily on the project's aims that shall be reached at its end. Consequently, answers to questions such as what is desired by the customer and what is requested by the development team to fulfill these desires, may determine the kind of required development artifacts at an early stage.

The user's manual represents one important development artifact because it is a powerful tool to summarize the final system's behavior, interaction models and capacity. The usage of paper prototypes serves as basis for a first draft of the user's manual. Its writing and the integration of screenshot mock-ups can be arranged in an early project phase. This preliminary user's manual structured in form of use cases acts as requirements specification for design and development [9, 10]. A stepless improvement will be reached from one iteration to another. As it is maintained by designers and developers, requested changes in specified use cases can be directly provided in the user's manual. So, the user's manual is created before or in parallel to the development work and always up-to-date. Additionally, this specification serves as basis for developing test scenarios and conducting user tests. After implementation, the screenshot mock-ups are replaced with real, final application screenshots. Hence, a minimum of required artifacts that have to be maintained are software and user's manual—a principle fact of agile software development.

Recommendation (R10). *Agility means flexibility and prioritization—but not cockaigne for incessant changes.* Both agile development and user-centered approach tempt to generate ideas and change requirements as desired. This possibility for incessant changes often carries cold sweat on the developer's forehead. Hence, these approaches may be regarded as annoyance to a well-ordered developer's life. As side effect to flexibility reached through agility, the development team may be plunged into troubles. Consequently, it is extremely important to prioritize features in cooperation with the product manager at the customer's site and to communicate clearly impact of changes on people, time and money. The fact that features and requirements are prioritized assumes that the product manager participates at iteration planning meetings. These meetings determine the development team's tasks and actions for the oncoming iteration phases. Furthermore, the prioritization of features allows to support feature-by-feature development and to establish small feature teams. A single team represents a group of designers and developers who altogether care for design, implementation and evaluation of particular features at a particular time. Besides, this procedure is evident for practicing agile development without declining in chaos. Hence, it requests for sophisticated, well-structured requirements information, management and planning of forthcom-

ing iterations. These are challenges, which have to be met by both the customer and the development institution.

6 Conclusion

When adopting an agile requirements process as important part of an overall agile software development process in a domain, where commonly formal processes are known and used, you are faced with a number of challenges. In this paper, we presented those challenges we were faced with when using an agile development process in a three years multidisciplinary software development project in the insurance domain. The challenges along with lessons learned in form of recommendations which we want to share within the RE community are summarized in Table 1. Challenges are written in text, as they are intended as index for the reader. For a detailed description of the challenges, we refer to Section 4. The recommendations associated with a specific challenge are abbreviated with *Rx*—for descriptions of recommendations, we refer to Section 5.

Challenge	Recommend.
(C1) Traditional processes at the customer's site	R1, R4, R8, R10
(C2) Misunderstandings of prototypes and final product	R2, R8, R9
(C3) Balancing management, workflow and technology tier	R1, R3, R5, R8, R10
(C4) Geographically distributed offices of the customer	R3, R5, R8
(C5) Diverse stakeholder interests	R3, R5, R6, R8, R10
(C6) Limited availability of field workers	R3, R8
(C7) Distributed development team	R3, R4, R6, R7, R8, R9

Table 1. Challenges and recommendations how to meet them.

To come back to Alexander et al. [7], we think that collaborations with research companies are also valuable for domains, where internal regulations and standards are used. Especially under conditions, when a company needs to get methodological know-how or also knowledge about state-of-the-art technologies.

7 Acknowledgements

The authors gratefully acknowledge support by the Austrian Government, the State of Upper Austria, and the Johannes Kepler University Linz in the framework of the K*plus* Competence Center Program.

References

[1] Agile alliance. http://www.agilealliance.org/. Last visited: February 2006.

[2] AUVA: Austrian social insurance for occupational risks. http://www.auva.at/. Last visited: June 2006.

[3] It-bvm: Bundesvorgehensmodell. http://www.bv-modell.at/. Last visited: June 2006.

[4] Manifesto for agile software development. http://www.agilemanifesto.org/. Last visited: June 2006.

[5] Microsoft solutions framework. http://msdn.microsoft.com/vstudio/teamsystem/msf/. Last visited: June 2006.

[6] V model: Development standard for it-systems of the federal republic of germany. http://www.v-modell.iabg.de/. Last visited: June 2006.

[7] I. Alexander, S. Robertson, and N. Maiden. What influences the requirements process in industry? A report on industrial practice. In *Proc. 13th IEEE International Conference on Requirements Engineering*, pages 411– 415, 2005.

[8] S. W. Ambler. Agile requirements modeling. http://www.agilemodeling.com/essays/agilerequirements.htm. Last visited: June 2006.

[9] D. M. Berry, K. Daudjee, J. Dong, I. Fainchtein, M. A. Nelson, T. Nelson, and L. Ou. User's manual as a requirements specification: case studies. *Requir. Eng.*, 9(1):67–82, 2004.

[10] T. DeMarco. *The Deadline*. Dorset House, New York, 1997.

[11] J. Dick and J. Chard. The systems engineering sandwich: Combining requirements, models and design. White paper, Telelogic, February 2004.

[12] J.A.Gulla. Understanding requirements in enterprise systems projects. In *Proc. 12th IEEE International Conference on Requirements Engineering*, pages 176– 185, 2004.

[13] E.-M. Melchior. User needs analysis: A precondition for validation planning. In *Inform Workshop*, Athens, Greece, 2003. [online, last visited: June 2006] www.vnet5.org/pub/inform-workshop-athens.html.