Some Findings Concerning Requirements in *Agile* Methodologies

Pilar Rodríguez, Agustín Yagüe, Pedro P. Alarcón, and Juan Garbajosa

Technical University of Madrid (UPM)
SYST Research Group
E.U. Informatica. Ctra. Valencia Km. 7. E-28031 Madrid
prodriguez@syst.eui.upm.es, agustin.yague@upm.es,
{pedrop.alarcon,jgs}@eui.upm.es

Abstract. Agile methods have appeared as an attractive alternative to conventional methodologies. These methods try to reduce the time to market and, indirectly, the cost of the product through flexible development and deep customer involvement. The processes related to requirements have been extensively studied in literature, in most cases in the frame of conventional methods. However, conclusions of conventional methodologies could not be necessarily valid for Agile; in some issues, conventional and Agile processes are radically different. As recent surveys report, inadequate project requirements is one of the most conflictive issues in agile approaches and better understanding about this is needed. This paper describes some findings concerning requirements activities in a project developed under an agile methodology. The project intended to evolve an existing product and, therefore, some background information was available. The major difficulties encountered were related to non-functional needs and management of requirements dependencies.

1 Introduction

Software industry is facing the fact that time to market is progressively becoming shorter. Agile approaches appeared as an attractive alternative to adapt the development to the unavoidable market changes, characterized by a continuous dynamism and variability [1]. Agile methods are suitable when the customer needs are quickly emerging and changing [2,3]. Their popularity is growing as they are able to better meet customer needs, improved quality software, faster time to delivery and lower development cost [4]. Assessments of agile in relation with other process models can be found in literature [5,6,7].

The experience that is being obtained from scaling up agile process models to large industrial projects and organizations [8][1] is showing us a radical breach between agile and other more conventional or traditional approaches. Agile process models, differently from more conventional software engineering process models, are structured into values, principles and practices [9,10]. As reported in [8] one

[1] This article develops Katti Vilki's keynote presentation at *Agile* 2008 Conference.

F. Bomarius et al. (Eds.): PROFES 2009, LNBIP 32, pp. 171–184, 2009.

of the reasons for this breach can be understood by the required application of agile values and principles to large projects and organizations; and not so much by the already well known practices such as continuous integration, integrated testing, or incremental delivery.

As it is nowadays accepted, the product quality is particularly dependent on how requirements engineering practices have been performed [11,12]. In [13,14] the differences between requirements specification in conventional and agile approaches are analyzed. Conventional methodologies are focused on *anticipation abilities* and can be termed as *plan based* [15,16] because these process models are defined in such a way that the later an error is discovered, the more expensive will be to correct it. They intend to identify a complete set of requirements in the requirement phase, what is always difficult to achieve. Once requirement phase is ended changes are always regarded as negative. Defining this complete set of requirements is essential for the soundness of the project, and if the problem domain is not well defined, this will affect negatively to the rest of the project [17,18,19]. As opposed to this, agile methods perceive each change like a chance to improve the system and increase the customer satisfaction. So, *responding to change over following a plan*[9] is one of the agile values. Agile teams do not try to avoid changes but try to understand what is behind them, seek to *embrace* them; the resulting set of requirements, after introducing a change, will be evaluated and rated searching for those requirements that will deliver the highest value to the customer. Therefore, change is considered as a normal and characteristic condition of software development.

One of the main aims of agile methods is to reduce the cost caused by these changes in requirements simplifying the requirements management and documentation tasks. Agile methods promote a fast and continuous communication between customers and development team. Face to face communication and frequent feedback are the most significant practices concerning to requirements engineering in these approaches [20]. The definition of tasks related to requirements is very often kept informal in agile approaches. Therefore, although there are evidences of the advantages that agile methodologies provide in small-scale projects, it is still difficult to scale to large projects applying among others the principle *responding to change over following a plan*.

Being Agile a relative young process model, there are few studies with relevant results about the elicitation and management of requirements. However, a recent survey [4] points out that inadequate project requirements and instability of requirements are among the important limitations of agile methods currently. Other papers, such as [20,13,21,22,23,24] report some problems in this area but do not analyze them in depth. Some of the open issues in agile methodologies concern elicitation of non-functional requirements and requirements documentation tasks.

In practice there are not studies that compare empirical results of agile and conventional projects referred to the same product. It is clear that it would be expensive to have two teams developing the same product. However in our case we had the opportunity to monitor the agile evolution of an existing product,

TOPENprimer, developed initially following a conventional approach. The existing requirements specification had been performed in compliance with IEEE[2] requirements specification standard 830-1998[25]. This was a good opportunity to get a better understanding of how Agile manages customer needs. That is how we were able to isolate specific requirements, understand the impact of missing requirements that were not identified at the supposedly appropriate moment of the agile development process. The study was performed considering the background on qualitative methods presented in [26].

The remainder of the paper is organized into four sections: Section 2 discusses related work about requirements engineering in agile approaches. Section 3 describes the case study in which the work is based and the process used in the development. Section 4, illustrates the identified issues with specific examples. Section 5 provides a reflection on the implications of the identified issues and possible correction mechanisms. Finally, Section 6 summarizes the findings and elaborate on future work.

2 Background and Related Work

Although some authors assert that agile methodologies are just old wine in new bottles [3], other studies show that product development in agile environments is very different to that in conventional environments [11,13,14,28]. Several experience reports, such as [29,30,31,32], describe success stories of using agile approaches. However, they do not usually provide enough context information or are merely a *lessons learned* report based on expert opinions do not focused on requirements. Others are designed to give recommendations and general rules for the agile methodologies use [3,33,34]. Requirements Engineering (RE) activities are considered critical to any software development process. It has been recognized that problems associated with the requirements area are among the major reason for software project failures [35,4]. The effort to explore and refine RE has grown up in the last years, as is pointed out by Nuseibeh in [11] and Cheng and Atlee in [12] in their studies about the current and the future in RE. However, there are still few studies about how real agile projects identify and manage the customer needs, and some authors suggest that the key issue is this [36]. Detractors argue that the quest for speed in software development may have the undesirable effect of weakening principles of purposefulness, appropriateness and truthfulness [37]. In contrast, current studies begin to identify and give solutions to existing problems. For example, in [21] to make an explicit requirements stage with customer is proposed or in [36] to add a conventional requirements stage. Araujo proposes to incorporate aspect orientation concepts in [38], in [39] it is proposed to deal with crosscutting requirements and in [40] to establish traceability. Other studies such as [22] are focused on giving high-level recommendations about identification and definition of customer needs in

[2] IEEE: Institute of Electrical and Electronics Engineers, Inc.

[3] Adapted from "Is Extreme Programming Just Old Wine in New Bottles: A Comparison of Two Cases" [27].

agile. In [41] the result of an experiment about the application of Requirements Interaction Management (RIM) process is showed. This study proposes changes in the agile requirement process, particularly in eXtreme Programming. Other publications, such as [20,13] identify some of the presented aspects in this paper but without going into them and point out the need to explicitly consider non-functional requirements management in Agile. However, none of these studies had the opportunity of compare the result of an agile and a conventional project referred to the same product as it is the case of this work. And finally, several studies such as [42,41,43,44] have been focused on interaction requirements and the conflicts related to this interaction. However, they are mostly focused on conventional methodologies.

3 Case Study: From TOPENprimer to TOPENbiogas

In this section we will provide a description of the case study in which the work is based. Subsection 3.1 describes the features of the product that has been evolved. The objective is to describe the project scope. In subsection 3.2, the used process is briefly described, focusing on the activities about customer needs management. Finally, in subsection 3.3 a list of some features existing in the initial product that were dropped in result product is presented; also a list of new features is included.

3.1 The Evolution Product Description

The case study was focused on the evolution of *TOPENprimer*. TOPENprimer was developed under a conventional methodology. It is based on the TOPEN (*Test Operation ENvironment*) architecture [45], that defines a domain specific environment for testing, monitoring and operating complex systems. TOPEN architecture is made up of four distributed components: *Topen Engine* is the kernel architecture. *Mission Information Base*(*MIB*) contains the database and the business rules. *Gateway* is the element that interacts with the *System Under Test* (*SUT*). And, finally, *TOE* is the user graphical interface. TOPEN follows a software product line approach [45] and it is specially designed to be adaptable to different application domains with a limited cost. For this reason, the evolution to a new domain implied, in general, a well-identified number of changes. On the one hand, this limits the scope of the study but, on the other, makes the study manageable. However taking advantage of the agile approach no feature was taken for granted in advance. The project consisted in the required evolution of TOPENprimer to support a new application domain. The target application domain was a biogas power production plant that had to be tested and monitored. *TOPENbiogas* was the result product in this project. In parallel, a biogas power plant simulator was developed in order to validate TOPENbiogas before its deployment in the real plant. More details about the evolution project are available in [46]. Some features of the product scope are shown in table 1.

Table 1. Characteristics of initial product TOPENprimer

Contextual Factor	**Characteristic Product**	**TOPENprimer**
Structure	Architecture	Four distributed components
Size	System Code Lines	30667
	Number of classes	216
	Code Lines	MIB: 7779
	by Component	TopenEngine:8372
		Gateway: 907
		TOE: 13609
	Number of classes	MIB: 48
	by component	TopenEngine: 55
		Gateway: 10
		TOE: 103
Technical Factors	Programming Language	Java
	Communication	Sockets, RMI

3.2 The *Agile* Development Process Description

The work reported here has been carried out within *ITEA2 Flexi project* [47]. Scrum [48] was used as the management methodology as it widely extended and Flexi partners were familiar with it. The constant feedback loops constitute the core element of the methodology. The development process is divided into short iterations called sprints. Figure 1 shows the Scrum project cycle. The sprint starts with a *planning* and finishes with *review and retrospective* stages. Features to be implemented in the system are registered in an artifact called *Product Backlog* (PB). In our case each feature was defined of a simple and clear way in form of *User Story* [49], in business language and prioritized by business value. At the beginning of each sprint, the Product Owner decides which PB

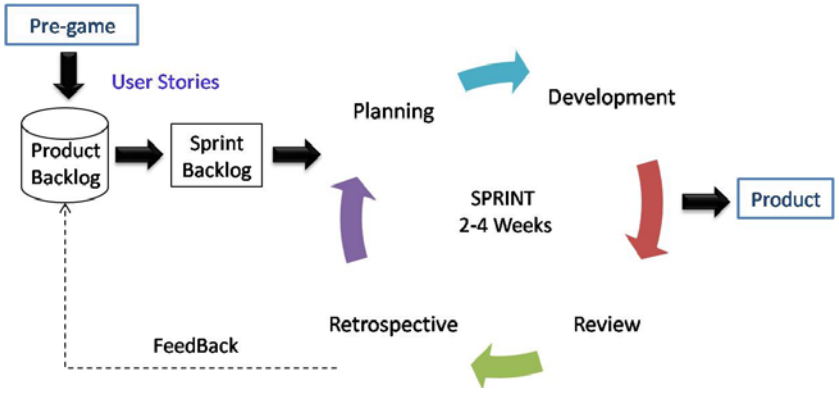

Fig. 1. *Agile* Development Model with SCRUM

items should be developed in that sprint. As can be seen in figure 1, there is not a specific task to pick up requirements. Pre-game is the most approximate stage because of its aims. In this stage, the scrum team, together with the customer, prepares a list of needs that the system should have in form of user stories.

3.3 Some New and Dropped Features

Some of the original TOPENprimer functionalities were modified. *Manager* facility was removed. *Managers* would have implemented operation views of the biogas plant; this feature is required to support cooperation of several stakeholders, e.g. an operator and an engineer. This could have been useful but it could be considered in a future upgraded version. A second issue was the Biogas plant visualization. Though the graphical user interface was important, it was agreed to postpone its implementation. A third issue was a Natural language facility. In TOPENprimer test/operation procedures are translated into natural language; the implementation of this feature was postponed. Finally, Operation Commands had some changes because some elements of the test/operation language (i.e. wait, for, repeat until, while, createNE or deleteNE) were not supported in the implemented version.

With respect to new features, a new kind of operation errors was considered because the complexity of the plant and its level of criticality were higher than that of slot machines. For instance, a gate cannot be close if it has not been opened before is a very critical restriction. Second, some internal identifiers were updated. This was transparent to the user, but implied a higher cost at MIB database level. Finally, command validation was done both at the real plant (simulator of real plant) and at TOPENbiogas. In TOPENprimer this validation was only done in the TOPEN environment.

4 Identified Issues in the Case Study

The Scrum methodology was tailored according to the specific project needs and the structure of the team. The project was developed in six sprints, fifteen days long each. The scrum team was made out of eight members (some of them with part-time). The customer provided the background documentation to define the User Stories and took part in the process, though a proxy customer was also used. This section describes some problems discovered during the study. Five fundamental issues were identified related to requirements working with Scrum methodology. In particular, the issues identified include requirements elicitation tasks, crosscutting requirements, derived requirements, granularity requirements and requirements documentation. These issues are not mutually exclusive.

4.1 Requirements Elicitation

Requirements elicitation activity intends to identify and understand customer needs. In agile approaches development tasks are not centered in a complete and well-defined set of requirements. User needs are incrementally elicited. In [14]

this closed relation with the customer is reported as very successful. However, we have found that it often happens that the customer is focused on issues on what the system has to do, forgetting other aspects, that may be become critical, such as the use of resources, maintenance, portability, safety, security or design. Most of these could be classified as non-functional requirements[4]. This happens because the customer usually does not have a vision of technical aspects. The problem is not so much how to express these requirements but the impact that may have on the product if they are not introduced at the right development stage.

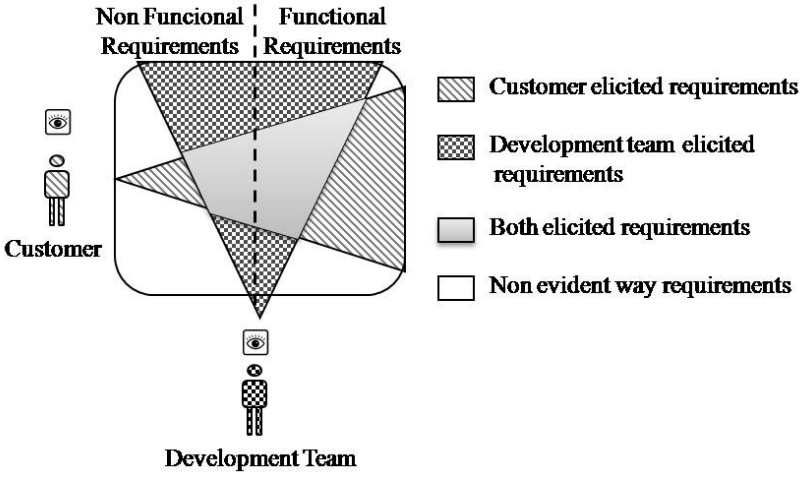

Fig. 2. System view from the team and customer

Actually it might be thought that most of the non-functional requirements should be known in the first stages of the development [13]. Although agile approaches contemplate an extensive use of refactoring techniques, the impact, e.g. to re-design a client-server architecture from a centralized could be dramatic. In our opinion, two main perspectives could be identified during the requirement elicitation: the customer view and the team view. Figure 2 shows it graphically. The customer perspective is functionality oriented leaving some product aspects out of its visibility, such as technical ones. At the other side, the development team perspective, depicted in the grid area, covers some requirements derived from the customer needs and some others of which the customer might not be aware of at all because of their nature. These include platform constraints, technical issues, and even development methodologies issues. As it can be shown in figure 2, there are some areas without any visibility. This is because at the beginning of the project all the requirements are not available.

[4] Within this paper, and referred to the experiment reported, non-functional includes what are called quality requirements for some authors "-ilities" and also design or other kind of requirements outside of functional.

4.2 Crosscutting Requirements

One of the features that had a strong impact on the project was the transversal nature of some requirements. This is the case of non-functional with respect to functional requirements, but non-functional requirements do not have the exclusivity of transversality. This is similar to the crosscutting concerns concept [38,50]. That is, non-functional requirements may be associated to many user stories. These crosscutting needs are difficult to break down into user stories such as in the case of safety. There is also no explicit way to express user stories interactions. A crosscut requirement is *spread* over several user stories, therefore, some tasks like planning, effort estimation or testing are affected. In the study case presented, this type of requirements has been managed under a new concept called System Story and that will be presented in the subsection 4.5. A specific example about this problem is shown in table 2. For example, if TOPENbiogas has to get access to the biogas plant locally and remotely; then all commands to be implemented have to consider this feature and planning, effort estimation and validation tasks are concerned. If it is identified too late it could have very serious implications on the product architecture what could delay unnecessarily get to an acceptable product.

Table 2. Example of Crosscutting Requirement

Formal Requirement Definition			
TOPEN environment can be accessed either locally or remotely			
System Story			
Id SS	**Who**	**What**	**Why**
SS6	Test Engineer	Access the environment locally or remotely	Operate and monitor Shredding Tank

4.3 Derived Requirements

Some required features could seem quite obvious and easy to obtain from the customer view. However, they could have an impact in the development tasks because some implicit related requirements are not still considered. In the study case, this type of hidden needs was classified as derived requirements (referring to those requirements that were derived from the analysis of other requirements). The communication protocols that use the TOPENbiogas commands are an example of this. These protocols are different if the environment works in local or remote access. In local, TOPENbiogas can check the components status *in situ* but not in remote. For this reason, the protocol has to be redefined to support other additional information when TOPENbiogas is working in remote access, as is shown in the table 2.

4.4 Granularity

Some user needs can be required at a lower level of detail. This happens not only in agile, of course, but in not conventional approaches the impact can be

lower as long as a long and detailed requirement process takes place. The issue in agile is to minimize the impact in case a requirement has to be split into lower granularity level ones. This is the case, for example, of the variables that are used to monitor the *Shredding Tank*, one component of the biogas plant. In a first iteration, the Shredding Tank was considered as the component to monitor. However, as the project went on, lower granularity variables, that have to be monitored too, appeared. The features of these variables affected the operation commands format that originally was defined in a too simplistic way. The result was to have to re-implement all the components. A lot of work was, probably unnecessarily, lost.

4.5 Customer Needs Documentation in Form of Stories

Finally, we found an important problem when we tried to represent some customer needs as user stories, which were already known in the initial product TOPENprimer. Those TOPENprimer requirements classified as functional could be written in user stories without problems. The problem appeared when we tried to include some needs such as the required database management system or the response time of TOPENbiogas. We found difficulties because the inclusion of features classified by conventional methodologies as non-functional, in the widest sense of this term, is not clearly defined in agile methodologies. We tested different solutions along the development. One was that these needs were included into user stories themselves. We considered this alternative because user stories describe features required by user and, anyway, non-functional requirements are special user expectations. However, according to Kassab [51], non functional requirements management is different to functional. Besides, many non-functional requirements often concern multiple user stories. In our project a new concept called System Story was used. System Stories have been defined as *"an added element to Agile methodologies that is used to collect any feature that customer/stakeholders want the system have related to non-functional requirements that could not be allocated in user stories"*

5 Discussion

The results achieved in the previous section show that, in agile methodologies, customer requirements elicitation and management require further maturation [39]. Therefore effectiveness can improve in the future. This section presents a discussion and some analysis of the previous results.

5.1 User Stories Interaction

User stories represent product needs that are defined and implemented in reduced time slot. Agile teams manage a high number of user stories, that grows up during the development duration, e.g. the Product Backlog in Scrum or the analog element in other agile methods is a dynamic artifact. As consequence of

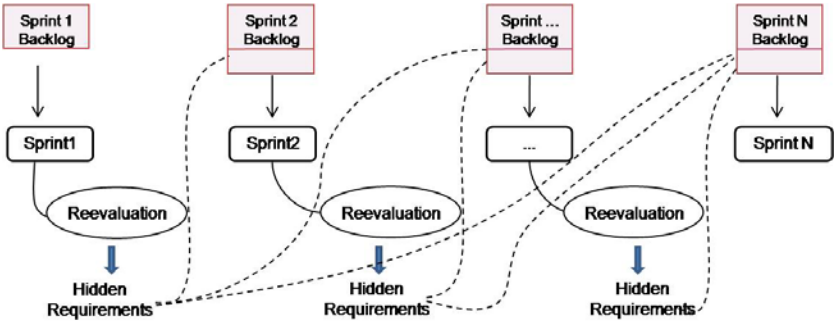

Fig. 3. Proposed life cicle for an *Agile* development

it, and from our own experience, the Product Backlog management is a complex task in agile methodologies like SCRUM. To consider that each user stories can be implemented independently of others is an error according to our experience. Several studies such as [42,41,43,44,17] have considered interaction of requirements and the conflicts related to this interaction. Most of the problems identified in section 4 are derived from these implicit requirement interactions what implies an overload to Product Backlog management.Although communication of team members is one of the principles of the Agile manifesto, some specific mechanisms to manage user stories dependencies should be advisable.

Table 3. Examples of User Story

User Story			
Id US	**Who**	**What**	**Why**
US31	Test Engineer	To change the shredding speed of the Tank	To operate Shredding Tank
US40	Test Engineer	To receive alert of Tank over-temperature	To monitor Shredding Tank

5.2 A Way to Review Stages

It seems reevaluation after each sprint should include not only well identified needs, but also other requirements such as crosscutting or derived requirements. Obviously the risk is loosing agility. In the case study we use the revaluation and re-prioritization of requirements stage at the end of each sprint to evaluate user stories that involve functional requirements from the perspective of potential non-functional requirements that are usually identified in a less obvious way. Figure 3 shows the proposed process. An example in our case study was in the Gateway component. It didnot appear in user stories because it is transparent to the user but was discovered in a revaluation stage. This component had to be completely redesigned and implemented to be adapted to the new communication protocol of the biogas plant.

5.3 Managing Non-functional Needs

As it has been shown in the study case, non-functional requirements management is one of the tasks that causes more problems in agile methodologies and it have not been still found a right solution. There are two tendencies related to this problem. On the one hand, an important agile methodologies sector thinks that user stories are able to represent any system need, both functional and non functional, and they do not consider a possible needs classification in agile approaches. On the other hand, there is an increasingly number of studies that find many difficulties to deal all requirements in the same way. They think that all customer needs are not equal and, therefore, it is necessary to distinguish some requirements be-cause their importance or management is different. For example, Bostrom et al. in [52] make a differentiation with security requirements suggesting Abuser Stories and Security-related User Stories to consider these needs. In the case study presented, we have found numerous problems to deal all needs equal, mainly management problems, and we have chosen to make different between functional and non-functional needs appointing the concept to System Story (see section 4.5)

6 Conclusions and Future Work

This paper presents some finding for requirements processes. These findings might be currently limiting the success of agile approaches. Elicitation and management of customer needs, specially non-functional, is an issue that requires further research; to get a better understanding of the inner relation between functional and non functional may yield in improved Agile approaches. Requirements dependencies is another important issue underlying many identified problems in this work, such as the management of crosscutting or derived requirements. These identified issues may be also relevant in conventional processes but this paper is an attempt to stress that they may be more critical for Agile processes. Read in other way, Agile processes may get a higher benefit if the research community progresses within this direction. The work planned for the future is dealing with gaining a better empirical knowledge combined with formal approaches. Another issue is studying how improved team cooperation can help in situations in which the mentioned issues come up.

Acknowledgements

The work reported has been partially sponsored by the Spanish MEC and MICYT under OVAL/PM TIC2006-14840 and FLEXI FIT-340005-2007-37 (ITEA2 6022). Besides, we would like to express public gratitude to BiogasFuelCell for their help in the application domain and Answare-tech, a partner in FLEXI. We are also grateful to the rest of the team: A. Espinoza, G. Rueda, J. Pérez, J. Díaz, A. Gómez and R. Cavero.

References

1. Boehm, B.: A view of 20th and 21st century software engineering. In: ICSE 2006: Proceedings of the 28th international conference on Software engineering, pp. 12–29. ACM, New York (2006)
2. Lindvall, M., Basili, V.R., Boehm, B.W., Costa, P., Dangle, K., Shull, F., Tesoriero, R., Williams, L.A., Zelkowitz, M.V.: Empirical findings in agile methods. In: Wells, D., Williams, L. (eds.) XP 2002. LNCS, vol. 2418, pp. 197–207. Springer, Heidelberg (2002)
3. Nerur, S., Mahapatra, R., Mangalaraj, G.: Challenges of migrating to agile methodologies. Commun. ACM 48(5), 72–78 (2005)
4. Vijayasarathy, L.R., Turk, D.: Agile software development: A survey of early adopters. Journal of Information Technology Management 19(2) (2008)
5. Boehm, B.W., Turner, R.: Balancing agility and discipline: Evaluating and integrating agile and plan-driven methods. In: ICSE, pp. 718–719. IEEE Computer Society Press, Los Alamitos (2004)
6. Boehm, B.W., Turner, R.: Management challenges to implementing agile processes in traditional development organizations. IEEE Software 22(5), 30–39 (2005)
7. Larman, C., Basili, V.R.: Iterative and incremental development: A brief history. Computer 36(6), 47–56 (2003)
8. Vilki, K.: Juggling with the paradoxes of agile transformation. Flexi Newsletter 2(1), 3–5 (2008)
9. Beck, K., Beedle, M., van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R.C., Mellor, S., Schwaber, K., Sutherland, J., Thomas, D.: Manifesto for agile software development (2001)
10. Beck, K., Andres, C.: Extreme Programming Explained: Embrace Change, 2nd edn. Addison-Wesley Professional, Reading (2004)
11. Nuseibeh, B., Easterbrook, S.: Requirements engineering: a roadmap. In: ICSE 2000: Proceedings of the Conference on The Future of Software Engineering, pp. 35–46. ACM Press, New York (2000)
12. Cheng, B.H.C., Atlee, J.M.: Research directions in requirements engineering. In: FOSE 2007: 2007 Future of Software Engineering, Washington, DC, USA, pp. 285–303. IEEE Computer Society Press, Los Alamitos (2007)
13. Paetsch, F., Eberlein, A., Maurer, F.: Requirements engineering and agile software development. In: WETICE 2003: Proceedings of the Twelfth International Workshop on Enabling Technologies, Washington, DC, USA, p. 308. IEEE Computer Society, Los Alamitos (2003)
14. Sillitti, A., Ceschi, M., Russo, B., Succi, G.: Managing uncertainty in requirements: A survey in documentation-driven and agile companies. In: METRICS 2005: Proceedings of the 11th IEEE International Software Metrics Symposium, Washington, DC, USA, p. 17. IEEE Computer Society, Los Alamitos (2005)
15. Miler, R.: Managing Software or Growth without fear, control, and the manufacturing mindset. Addison-Wesley Professional, Reading (2003)
16. Boehm, B.W.: Software Engineering Economics. Prentice-Hall Advances in Computing Science & Technology Series. Prentice Hall PTR, Englewood Cliffs (1981)
17. Damian, D., Chisan, J.: An empirical study of the complex relationships between requirements engineering processes and other processes that lead to payoffs in productivity, quality, and risk management. IEEE Trans. Softw. Eng. 32(7), 433–453 (2006)

18. Damian, D., Chisan, J., Vaidyanathasamy, L., Pal, Y.: Requirements engineering and downstream software development: Findings from a case study. Empirical Softw. Engg. 10(3), 255–283 (2005)
19. Basili, V.R., McGarry, F.E., Pajerski, R., Zelkowitz, M.V.: Lessons learned from 25 years of process improvement: the rise and fall of the nasa software engineering laboratory. In: ICSE 2002: Proceedings of the 24th International Conference on Software Engineering, pp. 69–79. ACM, New York (2002)
20. Cao, L., Ramesh, B.: Agile requirements engineering practices: An empirical study. IEEE Software 25(1), 60–67 (2008)
21. Grünbacher, P., Hofer, C.: Complementing xp with requirements negotiation. In: Proceedings 3rd Int. Conf. Extreme Programming and Agile Processes in Software Engineering, pp. 105–108. Springer, Heidelberg (2002)
22. Eberlein, A., Leite, J.: Agile requirements definition: A view from requirements engineering. In: International Workshop on Time-Constrained Requirements Engineering, Essen, Germany (September 2002)
23. Dyba, T., Dingsoyr, T.: Empirical studies of agile software development: A systematic review. Information and Software Technology 50(9-10), 833–859 (2008)
24. Neill, C.J., Laplante, P.A.: Requirements engineering: The state of the practice. IEEE Softw. 20(6), 40–45 (2003)
25. IEEE: IEEE Std 830-1998: IEEE Recommended Practice for Software Requirements Specifications (1998)
26. Seaman, C.: Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering 25(4), 557–572 (1999)
27. Merisalo-Rantanen, H., Tuunanen, T., Rossi, M.: Is extreme programming just old wine in new bottles: A comparison of two cases. J. Database Manag. 16(4), 41–61 (2005)
28. Ceschi, M., Sillitti, A., Succi, G., De Panfilis, S.: Project management in planbased and agile companies. IEEE Software 22(3), 21–27 (2005)
29. Sutherland, J.: Inventing and reinventing scrum in five companies (2001), `http://www.agilealliance.org/system/article/file/888/file.pdf` (accesed, May 2008)
30. Schwaber, K.: Agile Project Management With Scrum. Microsoft Press, Redmond (2004)
31. Mann, C., Maurer, F.: A case study on the impact of scrum on overtime and customer satisfaction. In: ADC 2005: Proceedings of the Agile Development Conference, Washington, DC, USA, pp. 70–79. IEEE Computer Society, Los Alamitos (2005)
32. Capiluppi, A., Fernandez-Ramil, J., Higman, J., Sharp, H.C., Smith, N.: An empirical study of the evolution of an agile-developed software system. In: ICSE 2007: Proceedings of the 29th international conference on Software Engineering, Washington, DC, USA, pp. 511–518. IEEE Computer Society, Los Alamitos (2007)
33. Baker, S.: Formalizing agility, part 2: how an agile organization embraced the cmmi. In: Agile Conference, p. 8 (July 2006)
34. Baker, S.W., Thomas, J.C.: Agile principles as a leadership value system: How agile memes survive and thrive in a corporate it culture. In: AGILE 2007: Proceedings of the AGILE 2007, Washington, DC, USA, pp. 415–420. IEEE Computer Society, Los Alamitos (2007)
35. Zowghi, D., Paryani, S.: Teaching requirements engineering through role playing: lessons learnt. In: Zowghi, D., Paryani, S. (eds.) Proceedings. 11th IEEE International Conference on Requirements Engineering, pp. 233–241 (September 2003)

36. Nawrocki, J.R., Michal Jasi, n., Walter, B., Wojciechowski, A.: Extreme programming modified: Embrace requirements engineering practices. In: RE 2002: Proceedings of the 10th Anniversary IEEE Joint International Conference on Requirements Engineering, Washington, DC, USA, pp. 303–310. IEEE Computer Society, Los Alamitos (2002)
37. Pinheiro, F.A.C.: Viewpoints: Requirements honesty. Requir. Eng. 8(3), 183–192 (2003)
38. Araujo, J., Ribeiro, J.: Towards an aspect-oriented agile requirements approach. In: Eighth International Workshop on Principles of Software Evolution, pp. 140–143 (September 2005)
39. Ribeiro, J.C., Araujo, J.: Asporas: A requirements agile approach based on scenarios and aspects. In: Second International Conference on Research Challenges in Information Science. RCIS 2008, pp. 313–324 (June 2008)
40. Lee, M.: Just-in-time requirements analysisthe engine that drives the planning game. In: Proc. 3rd Intl. Conf. Extreme Programming and Agile Processes in Software Eng. (XP 2002), pp. 138–141 (2002)
41. Woit, D.M.: Requirements interaction management in an extreme programming environment: a case study. In: ICSE 2005: Proceedings of the 27th international conference on Software engineering, pp. 489–494. ACM, New York (2005)
42. Robinson, W.N., Pawlowski, S.D., Volkov, V.: Requirements interaction management. ACM Comput. Surv. 35(2), 132–190 (2003)
43. Shehata, M., Eberlein, A., Fapojuwo, A.: Using semi-formal methods for detecting interactions among smart homes policies. Sci. Comput. Program. 67(2-3), 125–161 (2007)
44. Kim, M., Park, S., Sugumaran, V., Yang, H.: Managing requirements conflicts in software product lines: A goal and scenario based approach. Data Knowl. Eng. 61(3), 417–432 (2007)
45. Magro, B., Garbajosa, J., Perez, J.: A software product line definition for validation environments. In: 12th International Conference on Software Product Line, pp. 45–54 (September 2008)
46. Rodriguez, P., Yague, A., Alarcon, P., Garbajosa, J.: Metodologias agiles desde la perspectiva de la especificacion de requisitos funcionales y no funcionales. In: 13th Conference on Software Engineering and Databases, JISBD 2008 (2008)
47. The Flexi Research Project: Itea 2 flexi
48. Schwaber, K., Beedle, M.: Agile Software Development with Scrum. Prentice Hall PTR, Upper Saddle River (2001)
49. Cohn, M.: User Stories Applied: For Agile Software Development. The Addison-Wesley Signature Series. Addison-Wesley Professional, Reading (2004)
50. Murphy, G.C., Walker, R.J., Baniassad, E.L.A., Robillard, M.P., Lai, A., Kersten, M.A.: Does aspect-oriented programming work? Commun. ACM 44(10), 75–77 (2001)
51. Kassab, M., Daneva, M., Ormandjieva, O.: Scope management of non-functional requirements. In: 33rd EUROMICRO Conference on Software Engineering and Advanced Applications, pp. 409–417 (August 2007)
52. Boström, G., Wäyrynen, J., Bodén, M., Beznosov, K., Kruchten, P.: Extending xp practices to support security requirements engineering. In: SESS 2006: Proceedings of the 2006 international workshop on Software engineering for secure systems, pp. 11–18. ACM, New York (2006)