Agile Usability Patterns for User-Centered Design Final Stages

Ana Paula O. Bertholdo(✉), Fabio Kon, and Marco Aurélio Gerosa

Department of Computer Science, University of São Paulo, São Paulo, Brazil
{ana,fabio.kon,gerosa}@ime.usp.br

Abstract. The integration between Agile Methods and User-Centered Design (UCD) has been addressed by several authors in recent years. Nevertheless, a gap remains regarding a systematically consolidated description of agile usability practices for the final stages of UCD. Our aim is to describe agile usability practices based on the literature in the form of patterns, focusing on the UCD final stages, namely "Create Design Solutions" and "Evaluate Designs". A literature review was conducted to identify patterns of use of agile usability practices. The major results of the study presented here are the selection and classification of the usability practices for the UCD final stages within the agile community and their structured presentation in the form of patterns (Name, Context, Problem, Solution, and Examples). Presenting agile usability practices as patterns can increase their applicability; it facilitates the visualization of the similarities between the communities of UCD and Agile Methods and also presents the ideas more clearly to other communities that can benefit from using these patterns in their specific development contexts.

Keywords: Agile usability · Agile UCD · Agile UX · Best practices · Patterns

1 Introduction

The integration between agile methods and UCD has been addressed by several authors in recent years [1–6]. The challenge in combining these two methodologies focuses on finding the best way to carry out all the activities related to usability improvement, meeting users' real needs, designing and evaluating interfaces within agile environments, and having typical users involved in the activities.

There are several agile usability practices defined in the literature with different names, representing small variations of the same practice. The description of agile usability practices, in a common form, help to organize similar practices in a standardized way. It can also facilitate the visualization of similarities between the communities of UCD and Agile Methods.

In this context, we carried out earlier research leading to a description of usage patterns of agile usability practices in the early stages of UCD, namely

M. Kurosu (Ed.): HCI 2016, Part I, LNCS 9731, pp. 433–444, 2016.
DOI: 10.1007/978-3-319-39510-4_40

Identify Needs for Human-Centered Design, Specify Context of Use, and Specify Requirements [7]. However, a gap remains regarding a description of usage patterns of agile usability practices in the final stages of UCD.

This study aims to describe usability practices based on the literature using the patterns format, focusing on the UCD final stages, namely Create Design Solutions and Evaluate Designs, within agile environments. This paper complements our previous work [7] by answering the research question: What are the agile usability practices related to the final stages of UCD used?

2 Background

Sy [6] suggests that the aim of integrating UCD and agile methods [8,9] is breaking UCD stages down into the agile cycles size. To do so, UCD should be performed aiming at applying all the activities of the UCD cycle for each feature subset in agile iterations. The basis for many UCD methods is described in an international standard (ISO 13407: Human-Centered Design Process), which defines a general process for including human-centered activities in a development life-cycle, but does not specify specific methods. In this standard, when necessary to use a human-centered design process, four activities form the main cycle of work: (*i*) Specifying Context of Use; (*ii*) Specifying Requirements; (*iii*) Producing Design Solutions; (*iv*) Evaluating Designs.

In 2011, the systematic literature review on UCD and agile methods conducted by Silva et al. [4] presented 58 studies addressing this topic. Some of them depict an overall picture for integrating UCD and agile methods, such as Fox et al. [5] and Sy [6]. In 2014, the systematic literature review for agile processes and UCD integration conducted by Salah et al. [2] identified challenging factors that restrict Agile and UCD integration, exploring the proposed practices to solve them. This study included a total of 71 papers. In 2015, Brhel et al. [1] analyzed 83 relevant publications. The analysis resulted in five principles for user-centered agile software development.

With a more specific focus on practices related to the final stages of UCD, Silva et al. [10] noticed that it is difficult to perform traditional user testing sessions due to the tight schedules inherent to Agile. They described a set of practices used to evaluate software product usability. This paper aims at organizing the similar practices described in the literature in a pattern format and at providing a categorization for the patterns according to the UCD final stages.

3 Method

A literature review was conducted to find the usability practices used by the agile community. The search criterion was defined as follows:

– (("usability" OR "usability methods" OR "User Centered Design" OR "User eXperience" OR "Human-Computer Interaction" OR "Computer-Human Interaction") AND ("agile methods" OR "agile development" OR "eXtreme Programming" OR "Scrum" OR "agile")).

The filtering process consisted of: (*i*) Reading the title, (*ii*) Reading the summary, and (*iii*) Reading the complete study. Studies were included if they met the following criteria: (1) The study reported how usability practices were applied to agile communities in order to raise at least one of the following pieces of information for each practice: context, problem, or solution; (2) For separate studies using the same data, for example, dissertation and a paper, only the study with the most comprehensive report was included to avoid overloading a particular data set; (3) Studies written in English. For each phase, the studies that were not in accordance to the inclusion criteria were excluded.

4 Agile Usability Patterns for UCD Final Stages

The agile usability practices identified in our literature review were described in the following format: (*i*) Name; (*ii*) Context; (*iii*) Problem; (*iv*) Solution; and (*v*) Example, according to the definition of a pattern [11]. Following Alexander's definition [11], a pattern is a structured method of describing good design practices within a field of expertise. The selection of practices to become patterns considered only the practices used by at least three different cases.

The patterns were divided into categories according to the UCD stage [12] in which they are present, which facilitates understanding the goal of each pattern described. The selection and classification of the usability practices for UCD final stages within the agile community and presentation in the aforementioned format represent the major results of this research. This paper focuses only on the usability practices comprising the following steps of the UCD: (1) Create Design Solutions; and (2) Evaluate Designs.

4.1 Create Design Solutions

Pattern: *Low Fidelity Prototyping*

Context: Need to communicate and to validate an idea, with customers and team members, which is developed and refined quickly in agile environments. This communication is performed during the initial stage of an agile iteration and in meetings with team members and/or customers while the project development evolves.

Problem: Low fidelity prototyping for the key features of the entire system prior to development is common in traditional usability practices. This practice is incompatible with agile environments, in which the development cycle comprises a series of small incremental releases. Therefore, the problem is to get feedback from customers and typical users on the interface of a subset of features within an iteration and not for the entire system prior to the development. The main forces involved are:

– **Force 1**: Getting feedback from customers and users on the interface and the interaction flow in the early stages of the development cycle, when the features are not implemented.

– **Force 2**: Creating the interface design according to the features within an iteration and not for the key features of the entire system prior to the development, taking the "big picture" into account.

Solution: Low fidelity prototyping during the initial stage of an iteration and in meetings with team members and/or customers. The main difference between traditional low fidelity prototyping techniques and the ones used within agile environments is that prototypes are built for a subset of features of an iteration and not for the entire system prior to the development. The subset of features contains the key features of an iteration, for which improving understanding is necessary. Team members having boards with screen prototypes at the workplace. To specify the visual layout, teams that have interface design specialists create low fidelity prototypes as the basis for implementation.

Examples: For several authors, prototyping happens during the initial stages of the development and is used to evaluate usability both for inquiries and user testing [13,14]. Patton [15] defines the practice Prototype in low fidelity, informing that the prototypes only need to be good enough to understand, to learn, and to communicate quickly. After that, he says, they can be thrown away, since they are consumable and not deliverable for agile development.

Pattern: *High Fidelity Prototyping*

Context: Need to evaluate the interface and the interaction flow of a system during the development cycle, when the features have been implemented.

Problem: High fidelity prototyping for the key features of the entire system prior to development is common in traditional usability practices. This practice is incompatible to agile environments, in which the development cycle comprises a series of small incremental releases. Therefore, the problem is to get feedback from customers and users on the interfaces of features that have been implemented in each agile iteration and not for the key features of the entire system prior to the development. The main forces involved are:

– **Force 1**: Getting feedback from customers and users on the interface and on the interaction flow during the development cycle, when the features have been implemented.
– **Force 2**: Creating the interface design according to the features of an iteration and not for the key features of the entire system prior to the development, taking the "big picture" into account.

Solution: High fidelity prototypes are part of the system under development, which is evolving at each iteration, already with the working code. When the features have been implemented, they are evaluated as high fidelity prototypes which evolve depending on the development of iteration requirements. Online techniques such as sharing access to prototype versions on the Web are used. In an iteration, teams create the interface design and, at the subsequent iteration,

the team of developers writes the code implementing the design proposed in the previous iteration.

Examples: For Williams and Ferguson [16], prototypes evolve for high fidelity prototypes. Hussain [17] cites the use of high fidelity prototypes to conduct inquiries and usability tests with the customer. Six [18] states that: "... a user experience team that includes front-end Web developers can prototype or even build an application's actual user interface rather than writing detailed user experience design specifications".

Pattern: *Design Studio*

Context: After some UCD tasks for gathering requirements are performed, usually in the format of user interviews or observations, there is the need to create and to explore design versions that meet the users goals.

Problem: Creating alternative design versions for the key features of the entire system prior to development is a way to allow the innovation to be part of the development of traditional usability practices. However, in agile environments, the short time of each iteration prevents exploiting several design options. The main forces involved are:

- **Force 1**: Proposing innovative solutions within agile iterations, with less time to build different interface design ideas and interaction flows.
- **Force 2**: Proposing innovative solutions for features of the next iteration and not for the key features of the entire system prior to the development, taking the "big picture" into account.

Solution: Participants come up with several alternative design solutions for an interface during a pre-established time. When time is over, a discussion is performed to select a design for the system interface. In the Design Studio, each design sketch is presented to the team, which can involve developers and UCD specialists. Then, the team has some time for reflection and criticism. At the end of the studio, a design concept is defined and developed. The chosen design reflects the good parts of the ideas presented during the studio. After the studio, the user experience team creates usage scenarios from the results. Since the design studio makes it easier to understand what the system needs are, the development work may proceed immediately [19]. Members of the development team might be allocated for user interface design activities. Usually, design studio tasks are performed face-to-face with members gathered in a room to create design versions.

Examples: Dubakov [20] describes the Design Studio methodology as a simple and efficient way of having agile meetings. The author reports that there are several variations for the Design Studio, but he used a simple group containing the following 5 steps: (1) defining a problem; (2) individually brainstorming 5 ideas without exceeding 5 min per idea; (3) presenting and divide the ideas into categories for the team; (4) discussing positive and negative characteristics

of the ideas; and (5) selecting the interesting ideas and creating two versions for each final solution. According to Ungar et al. [19], design studio has four main points: (1) Research: Design Studio is guided by user research; (2) Design: also known as pre-work, where many projects and ideas are quickly created; (3) Studio: a one-day workshop to evaluate alternatives, make decisions and consolidate design and (4) Participants: a team composed of designers or user experience professionals who are willing to learn and to grow within the design process. Evans and Gothelf [21] describe that a design studio might generate several iterations about the design and that there are several variations of the studio which might be employed with good effects within agile processes.

Pattern: *Collaborative and Participative Design*

Context: After some UCD tasks for gathering requirements are performed, usually in the format of user interviews or observations, it is necessary to put the perspective of each participant involved with the development cycle, to include typical users in the interface design process, for improvement and refinement of systems requirements.

Problem: In traditional usability practices, the UCD team creates alternative design versions for the key features of the entire system prior to development such that the perspective of users and customers can be incorporated. However, when just the UCD team is involved with creating design solutions, the understanding is not complete for the entire team to build only what has the higher value for the business and for the end user. The main forces involved are:

- **Force 1**: Gathering the perspective of each participant for a design problem into agile iterations, with less time for each session.
- **Force 2**: Proposing design solutions for features of the next iteration and not for the key features of the entire system prior to the development, taking the "big picture" into account.

Solution: Sessions for creating design versions in a collaborative way, i.e., different stakeholders create designs together, showing their views in relation to the needs that have to be met by the user interface, and in a participative way, by involving system real users. The objective is to put the point of view of each participant to solve a design problem based on their needs in relation to the system and their own understanding of it. As in the Design Studio, this activity stimulates innovation. Also, it allows new requirements to be suggested and gathered during the sessions. The involvement of developers is very beneficial to the quality of the system, both internal and external. Additionally, it allows all team members to become involved with the stage of defining requirements. Collaborative design also contributes to creating a shared view among participants.

Examples: Govella [22] includes collaborative design in his list of strategies for the user experience, in which members of the design team may sketch, talk, and iterate to achieve a consensual idea of the design. Details only need to be recorded if they might be forgotten. The same happens if there is an error, but

all members understand the general view since they have worked together. For Beltrame [23], collaborative design is all about leaving developers do part of the ideation process and designers do part of the development process. Developers might help proposing good solutions, which are technically suitable, and avoiding conceptual failures due to technical limitations. Tyne [24] describes participative design in three main tasks: (1) sketching ideas, (2) presenting ideas to the team, and (3) criticizing presentation based on solutions. The process is repeated three times and the fidelity of scopes increases at each cycle.

4.2 Evaluate Designs

Pattern: ***Tests with Users***

Context: Need to obtain a spontaneous impression from the user about a version of the system.

Problem: Evaluating the system with real users is an essential practice in traditional usability practices. However, is not a trivial task to perform tests with users for each agile iteration with less time to run evaluations. The main forces involved are:

- **Force 1**: Evaluating the system with its real users from the early stages of the development process.
- **Force 2**: Conducting usability tests according to the functionalities developed in an iteration or agile sprint, taking the "big picture" into account.

Solution: Conducting usability tests with users employing the Think aloud protocol. The central idea is to provide tasks for the users to perform while using the system, aiming at evaluating if the tasks are easily performed or if they encounter difficulties discovering the right path, or even if they are unable to do the required task. The Think Aloud protocol aims at requesting users to say what they intend to do to complete the task, what they think about the interface, and what they thought about the steps they had to follow to accomplish the task successfully. While users are using the system, their impressions are gathered so as to analyze the results in relation to the accomplishment of the task. Usually, in agile teams, tests with users are conducted with the presence of project customers and involve usability specialists. When real users are involved, usability specialists also participate as test mediators. However, there are reports of projects that involved several members of the team as observers and they noticed that the developers' view benefited from watching how real users understand the system. Tests with users in agile teams are mostly laboratory-based, although there are reports of successful remote tests. Normally, developers incorporate results from tests with users into the system during the following iteration.

Examples: Expero [25] reports that when tests with users are conducted as part of the agile process, the tests should occur before or during the development and should have user tasks that take the "big picture" into account more than the

limited scope of the next release. Another unique aspect of tests with users within agile environments is to make sure that the changes recommended from user studies find a solution in an appropriate release. It might be the case of planning a release for the following months. Frequently, the development team changes characteristics and functionalities. There are several reports on tests with users functioning as the main tool for refining user interface prototype for the next iteration [4,5,16].

Pattern: ***Evaluation by Inspection***

Context: Need to evaluate the interface even at the initial stages of the system without exposing serious issues to the real users.

Problem: In traditional usability practices, inspecting interface is an important task to refine the system before introducing the features for customers or real users. However, in agile environments, the interface refinement should be performed for each agile iteration, in which time is shorter to execute inspection and to correct problems. The main forces involved are:

- **Force 1**: Evaluating system usage scenarios from the very beginning and frequently during development.
- **Force 2**: Not showing serious problems to real users.
- **Force 3**: Evaluating the usability of features of the current agile iteration.

Solution: Inspecting user interfaces, of low- or high-fidelity prototypes, through usability scenarios, rules, or heuristics. Usability specialists perform evaluation by inspection to verify whether the user interface under test meets the rules or follows expected paths for the given scenarios. Normally, two or three specialists execute it. First, each specialist individually analyzes and then what was found to consolidate results is discussed by all. This way, tests with users are conducted in interfaces that were already evaluated internally, without exposing (ideally) serious issues to the real users of the product. Specialists share results with the development team so that problems found may be corrected. Some teams report the involvement of developers and, in this case, they describe such participation as positive, since it facilitates the process of sharing usability knowledge with the team. Also, meetings for consolidating results found in each individual analysis are mostly laboratory-based. Usually, developers solve problems found during the evaluation by inspection in the same iteration or sprint.

Examples: Similar to what occurs during tests with users, the systematic review by Silva et al. [4] also raised user interface refinement as the main goal of evaluation by inspection, according to [13,16,19]. Obendorf and Finck [26] state it is possible to use scenarios to guide evaluation by inspection of paper prototypes.

Pattern: *RITE Method*

Context: Need to identify and to solve as many problems as possible and to check the efficacy of solutions as soon as possible.

Problem: In agile environments, there is not enough time for usability tests and correction of problems, for each iteration, using traditional usability methods, which leads to rare usability tests and the correction of few bugs. The main forces involved are:

- **Force 1**: Receiving feedback from corrections as close as possible to tests conduction.
- **Force 2**: Shortening the distance between finding usability problems and correcting them in the system.
- **Force 3**: Evaluating features within agile iterations, with less time to run tests and to correct bugs.

Solution: Identifying and solving problems found, by using tests with typical users of the system and checking how efficient the solutions are as soon as possible. Rapid Interactive Testing and Evaluation (RITE) emphasizes changes and quick verifications of their efficacy. Usability testing and the quickest correction of errors. Usually, usability specialists perform RITE and its results are shared with other team members to be evaluated.

Examples: This practice is present in the list of best practices published by the User eXperience Magazine, focusing on changes and verifications of their efficiency quickly [27]. For Patton [15], RITE is used to iterate the user interface before the development. In [28], RITE is defined as a variation of traditional usability tests, documented by Microsoft researchers in 2002, being credited to Medlock. In sum, by testing a design with five users on the first day, on the second day, the design is improved based on the feedback; it is tested again on the third day; on the fourth day, another iteration is conducted, and then the final design is tested on the fifth day with eight users. The authors also state that RITE is not always appropriate (if there are too many tasks, for example). However, whenever possible, it is highly recommended by them since it saves time, promotes collaboration within the team besides customer satisfaction.

Pattern: *Acceptance Tests*

Context: Need to check if the goals of real users and customers were met in a version of the system already with the working code.

Problem: Validating whether the needs of customers and typical users are being met by the features prior to development is common in traditional usability practices, which is incompatible with agile environments, which entail adaptive planning and iterative processes. In agile environments, this task should be performed keeping the documentation updated in relation to features that successfully or unsuccessfully meet the needs of users. The main forces involved are:

- **Force 1**: Making the validation of the system by customers and typical users from the very beginning and frequently during development.
- **Force 2**: Keeping the documentation updated according to already-implemented user needs.
- **Force 3**: Evaluating features within agile iterations, with less time to run tests.

Solution: Creating automatic acceptance tests using Test Driven Development (TDD), which is already part of agile methods. Tests are created with the customers or typical user participation by using standard formats similar to natural language, such that non-experts on programming are capable of describing user needs. By conducting automatic tests based on TDD, a report is generated showing the status of the defined tests. This practice helps validating the user needs and creates an updated documentation of everything the system does, which functionalities already have tests, and whether the tests are executed successfully.

Examples: Dyba et al. [29] describe automatic acceptance testing as a good practice to validate user interface design. The agile community acknowledged the importance of acceptance tests and built tools such as the Framework for Integrated Test (fit.c2.com), JBehave (jbehave.org), and Cucumber (cucumber.io) to help automate them. Automated testing is performed frequently, at least every day or several times a day. Manual tests with users, on the other hand, are normally conducted one iteration ahead. At the end of an iteration, many agile teams implement a system functioning in a testing environment, where automated tests of the system and tests with users are conducted. The team continues to develop version N+1 of the system while gathering reports on version N. Failure reports are treated as any other requirement: they are estimated, prioritized, and put into a requirement list to be treated in the future [30]. According to Rice [31], these tests should not be treated as "functional tests based solely on user requirements", since you are "likely to miss the same things in testing that were missed in defining the requirements".

5 Conclusion

According to Sy [6], all the activities of the UCD cycle should be performed for each subset of features in agile iterations. The patterns presented organize the practices described in the literature in a common format. Therefore, it is possible to select the patterns of each phase of the UCD cycle that best fit specific development environments and apply them within an agile iteration.

In addition, the patterns presented here can help understand how to integrate the practices of UCD and agile methods, allowing developers to visualize the similarities shared, and also the commonalities among other communities that could use the patterns in their contexts of development. We described a set of patterns of agile usability practices for UCD final stages extracted from the literature. The presentation of patterns in a format including name, context, problem, solution, and examples can increase their applicability.

Overall, the patterns described for the UCD stage - Create design solutions - have in common the fact of seeking knowledge sharing in order to facilitate understanding the requirements and the user interaction flow with the system. For the patterns described for the UCD stage - Evaluate designs - there is the need of user interface refinement for the following iteration. The goal is to evaluate designs during the development process, from early stages with prototypes in low fidelity. This work complements our previous study on the early stages of UCD [7]. Future studies may define an order of application of the patterns according to the stage of development in an agile environment while maintaining the DCU cycle for each agile iteration.

Acknowledgements. This research was supported by FAPESP, Brazil, proc. 2012/24409-2, and the European Comission, proc. 034763.

References

1. Brhel, M., Meth, H., Maedcher, A.: Exploring principles of user-centered agile software development: a literature review. Inf. Softw. Technol. **61**(C), 163–181 (2015)
2. Salah, D., Paige, R.F., Cairns, P.: A systematic literature review for agile development processes and user centred design integration. In: Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering, 13–14 May 2014
3. Silva da Silva, T., Silveira, M., Maurer, F., Hellmann, T.: User experience design and agile development: from theory to practice. J. Softw. Eng. Appl. **5**, 743–751 (2012)
4. Silva da Silva, T., Martin, A., Maurer, F., Silveira, M.: User-centered design and agile methods: a systematic review. In: Imperial College Robotics Society (ed.) Agile Conference (Agile) 2011, pp. 77–86 (2011)
5. Fox, D., Sillito, J., Maurer, F.: Agile methods and user-centered design: how these two methodologies are being successfully integrated in industry. In: Agile 2008, AGILE 2008 Conference, pp. 63–72 (2008)
6. Sy, D.: Adapting usability investigations for agile user-centered design. J. Usability Stud. **2**(3), 112–132 (2007)
7. Bertholdo, A.P.O., da Silva, T.S., de O. Melo, C., Kon, F., Silveira, M.S.: Agile usability patterns for UCD early stages. In: Marcus, A. (ed.) DUXU 2014, Part I. LNCS, vol. 8517, pp. 33–44. Springer, Heidelberg (2014)
8. Beck, K., Andres, C.: Extreme Programming Explained Embrace Change. Addison-Wesley Professional, Reading (2004)
9. Schwaber, K., Beedle, M.: Agile Software Development with Scrum, 1st edn. Prentice Hall PTR, Upper Saddle River (2001)
10. Silva Da Silva, T., Selbach Silveira, M., Maurer, F.: Usability evaluation practices within agile development. In: 2015 48th Hawaii International Conference on System Sciences (HICSS), pp. 5133–5142. IEEE (2015)
11. Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., King, I.F., Angel, S.: A Pattern Language: Towns, Buildings, Construction. Center for Environmental Structure Series. Oxford University Press, New York (1977)
12. Association, U.E.P.: What is user-centered design? January 2014

13. Fox, D., Sillito, J., Maurer, F.: Agile methods and user-centered design: how these two methodologies are being successfully integrated in industry. In: Proceedings of the Agile 2008, pp. 63–72 (2008)
14. Detweiler, M.: Managing ucd within agile projects. Interactions **14**, 40–42 (2007)
15. Patton, J.: Emerging best agile ux practice (2008). http://agileproductdesign.com/blog/emerging_best_agile_ux_practice.html
16. Williams, H., Ferguson, A.: The ucd perspective: before and after agile. In: AGILE 2007, pp. 285–290 (2007)
17. Hussain, Z., Milchrahm, H., Shahzad, S., Slany, W., Tscheligi, M., Wolkerstorfer, P.: Integration of extreme programming and user-centered design: lessons learned. In: Abrahamsson, P., Marchesi, M., Maurer, F. (eds.) Agile Processes in Software Engineering and Extreme Programming. LNBIP, vol. 31, pp. 174–179. Springer, Heidelberg (2009)
18. Six, J.M.: Integrating ux into agile development, April 2011. http://www.uxmatters.com/mt/archives/2011/04/integrating-ux-into-agile-development.php. Accessed Dec 2011
19. Ungar, J., White, J.: Agile user centered design: enter the design studio - a case study. In: CHI 2008 Extended Abstracts on Human Factors in Computing Systems. CHI EA 2008, pp. 2167–2178. ACM, New York (2008)
20. Dubakov, M.: Ux meets agile: design studio methodology, May 2011. http://www.targetprocess.com/blog/2011/05/ux-meets-agile-design-studio-methodology.html. Accessed Dec 2011
21. Evans, W., Gothelf, J.: Design studio and agile ux: process and pitfalls, November 2011. http://uxmag.com/articles/design-studio-and-agile-ux-process-and-pitfalls. Accessed Dec 2011
22. Govella, A.: Agile + ux: six strategies for more agile user experience (2008). http://www.thinkingandmaking.com/view/agile-ux-six
23. Beltrame, M.: Just married: user centered design and agile, May 2011. http://www.memibeltrame.ch/slides/. Accessed Dec 2011
24. Tyne, S.V.: User experience design in agile development (2011). http://www.slideshare.net/sdeconf/sdec-2011-uxagilesvt. Accessed Dec 2011
25. Enterprise E: Incorporating user-centered design into an agile development process, December 2011. http://experoinc.com/incorporating-user-centered-design-into-an-agile-development-process/. Accessed Dec 2011
26. Obendorf, H., Finck, M.: Scenario-based usability engineering techniques in agile development processes. In: CHI 2008 Extended Abstracts on Human Factors in Computing Systems, pp. 2159–2166. ACM, New York (2008)
27. Lu, C., Rauch, T., Miller, L.: Agile teams: best practices for agile development. vol. 9, no. 1, pp. 6–10 (2010)
28. Leggett, N.: User research findings - analyzing the user research segment - the power of doing it rite, July 2008. http://www.userresearchfindings.com/2008/07/power-of-doing-it-rite.html. Accessed Dec 2011
29. Dingsoyr, T., Dybå, T., Moe, N.B.: Agile Software Development - Current Research and Future Directions, 1st edn. Springer, Heidelberg (2010)
30. Ambler, S.W.: Introduction to agile usability user experience activities on agile development projects - user testing on an agile project (2009). http://www.agilemodeling.com/essays/agileUsability.htm#AcceptanceTesting. Accessed Dec 2011
31. Rice, R.W.: What is user acceptance testing? (2009). http://www.riceconsulting.com/articles/what-is-UAT.htm. Accessed Dec 2011