

Incorporating User Centered Requirement Engineering into Agile Software Development

Markus Duechting¹, Dirk Zimmermann², Karsten Nebe¹

¹ University of Paderborn C-LAB, Cooperative Computing & Communication Laboratory,
Fürstenallee 11, 33102 Paderborn, Germany
{markus.duechting, karsten.nebe}@c-lab.de

² T-Mobile Germany, Landgrabenweg 151,
53227 Bonn, Germany
dirk.zimmermann@t-mobile.de

Abstract. Agile Software Engineering approaches gain more and more popularity in today's development organizations. The need for usable products is also a growing factor for organizations. Thus, their development processes have to react on this demand and have to offer approaches to integrate the factor "usability" in their development processes. The approach presented in this paper evaluates how agile software engineering models consider activities of Usability Engineering to ensure the creation of usable software products. The user-centeredness of the two agile SE models Scrum and XP has been analyzed and the question of how potential gaps can be filled without losing the process' agility is discussed. As requirements play a decisive role during software development, in Software Engineering as well as Usability Engineering. Therefore, different User Centered Requirements that ensure the development of usable systems served as basis for the gap-analysis.

Keywords: Agile Software Engineering, Usability Engineering, User-Centered Requirements.

1 Traditional Software Engineering

The ambition of Software Engineering (SE) is the systematic and manageable development of Software, in order to make the development process more planable. Many existing Software Engineering Models (SE Models) manage the software development regarding costs, time and quality. A well-established and prevalent SE model is the Waterfall Model [12] introduced by Royce. The model consists of seven phases that are passed through sequentially. The results of each phase are captured in documents, which serve as milestones for further development activities of the next phase. The Waterfall Model offers an easy way to schedule and manage the software development, because of its sequential progress. It has been successfully applied in situations where requirements and workflows can be completely determined upfront.

Another common SE Model is the Spiral Model introduced by Boehm [2], which is an enhancement of the Waterfall Model. The Spiral Model is an iterative and incremental approach which provides a cyclic repetition of four phases. Each cycle of the spiral consists of four major activities and ends with a progress assessment, followed by a planning phase for the next process iteration. Additionally, a risk assessment is performed after each iteration. The iterative approach allows reacting adequately on changing requirements. This makes the process of developing software more manageable and minimizes the risk of failure, in contrast to the sequential SE Model.

2 Agile Software Engineering

A recently emerging trend in SE focuses on lightweight, so called agile models, which follow a different approach to software development. Agile models follow the idea of Iterative and Incremental Development (IID), similar to the Spiral Model mentioned above. But in contrast to Boehm's model, the iteration length is shorter in agile models. The iterations in the Scrum Model for instance, take 30 calendar days. Agile software development does not rely on comprehensive documentation and monolithic analysis activities; instead they are more delivery- and code-quality-oriented approaches. Through co-location of the development team the tacit knowledge among the team members compensates extensive documentation efforts. Agile models emphasize communication, and aspire towards early and frequent feedback through testing, on-site customers and continuous reviews.

The basic motivation behind agile and iterative development is to acknowledge that software development is similar to creating new and inventive products [9]. New product development requires the possibility for research and creativity. It is rarely possible to gather all requirements of a complex software system upfront and identify, define and schedule all detailed activities. Many details emerge later during the development process. This is a known problem within the domain of SE and the reason for many failed projects [9]. For this reason, agile models implement mechanisms to deal with changing requirements and other unforeseen incidents to plan, monitor and manage SE activities.

2.1 Scrum

Scrum is an agile and iterative-incremental SE model. Its development tasks are organized in short iterations, called **Sprints**. Each Sprint starts with a Sprint Planning meeting where stakeholders decide the functionality to be developed in the following Sprint. All requirements for a software system are collected in the **Product Backlog**. The Product Backlog is a prioritized list and serves as a repository for all requirements related to the product. However, the Product Backlog is not at any time a finalized document but rather evolves along with the product. In the beginning of a project the Product Backlog only contains high-level requirements and it becomes more and more precise during the Sprints. Each Backlog item has a priority assigned

to represent its' business value, and an effort estimation to plan the required resources to implement it. During the Sprint Planning, the **Scrum Team** picks high priority backlog items that they think are realistic for the next Sprint.

The Scrum Teams are small interdisciplinary groups of 7 to 9 people [13], which are self-organized and have full authority to determine the best way for reaching the Sprint Goals. There are no explicit roles defined within the Scrum Team. Scrum places emphasis on an emergent behavior of the team, meaning the teams develop their mode of cooperation autonomously. This self-organizing aspect supports creativity and high productivity [13]. The Scrum Team and its' manager - the **Scrum Master** – meet in a short, daily meeting, called **Daily Scrum**, to report progress, impediments and further proceedings. Every Sprint ends with a **Sprint Review** meeting, where the current product increment is demonstrated to project stakeholders.

2.2 Extreme Programming

Extreme Programming [1] is one of the established agile SE methodologies. Similar to Scrum, XP is an iterative-incremental development model. However, XP's iterations are even shorter than Scrum's. According to Beck the optimal iteration-length is somewhere between 1 and 3 weeks. XP adopts reliable SE techniques to a very high degree. Continuous reviewing is assured by **pair programming**, where two developers are sitting together at one workstation. XP also applies the common code ownership principle. All team members are allowed to make changes in code written by someone else when it is necessary. In addition, XP requires a user stakeholder to be on-site as a mean to gather early user feedback. The requirements in XP are defined by the customer in so called **User Stories**. Each story is a brief, informal specification of requirements. Similar to Scrum's Product- and Sprint Backlog, the User Stories have a priority and effort estimation assigned to it.

Before a new iteration starts, the User Stories are decomposed into more granular technical work packages. The literature about XP does not mention an explicit design phase, but highly emphasizes continuous refactoring and modeling. The functionality described in User Stories is converted into test cases. The simplest concept that passes the test is implemented. Development is finished, when all tests are passed.

3 User Centered Design

A recent trend can be observed, showing that usability criteria become a sales argument for products and the awareness for the need of usable systems is growing. But many software development projects are mainly driven by the SE model that is used.

Usability Engineering (UE) provides a wide range of methods and systematical approaches to support the user-centered development. These approaches are called Usability Engineering Models (UE Models), e.g. the Usability Engineering Lifecycle [10] or Goal-Directed Design [4]. Mayhew's UE process consists of three phases, which are processed sequentially. The first Phase is the Requirement Analysis,

followed by Design/Testing/ Development Phase and the Installation of the product. The process is iterative: Concepts, preliminary and detailed design are evaluated until all problems are identified and resolved.

In the Goal Directed Design Process of Cooper, several phases are passed through as well. During the **Research Phase**, qualitative research leads to a picture of how users do work in their daily work environment. During the **Modeling Phase** Domain Models and User Models (so called Personas) are developed that are then translated into a **Framework** for the design solutions, which is detailed in the **Refinement Phase**. These two models have much in common since they describe an idealized approach to ensure the usability of a software system, but they usually differ in the details. UE Models usually define an order of activities and their resulting deliverables. UE approaches often happen concurrently to the other software development activities, so there is an obvious necessity for integrating these two approaches, in order to permit the predictability of budgets, resources and timelines of the UE activities within software development.

4 Motivation

According to Ferre [7] basic conditions for integrating SE and UE are an iterative approach and active user involvement. The two agile SE models outlined above are iterative-incremental approaches that rely on a solid customer involvement. They even talk about user representatives as a special kind of customer stakeholders. The involved customer should at least have a solid knowledge of the user's domain and their needs. This raises the question, if and how Usability is ensured in an agile software development process in order to perform UE activities in a satisfying way. This paper discusses the user-centeredness of two agile SE Models and the question how potential gaps can be filled without losing the process agility.

When exploring the UCD Models described above, there is a commonality with the traditional SE Models. Both are strongly driven by phases and the resulting deliverables. However, documentation has a minor part in agile models. Due to their incremental approach and overlapping development phases there are no distinct phases, like e.g. Analysis, Design, Development and Validation, in agile SE Models. Without certain deliverables or activities there is a need for other criteria to allow an assessment of the user-centeredness of agile SE Models.

Requirements play a decisive role during the software development lifecycle, in both the SE and the UE domain. SE is mainly concerned with system requirements, while UE takes the user's needs into account. Requirements are measurable criteria and the elicitation, implementation and validation takes place in most approaches to software development. The approach of defining granular requirements allows to look at activities independent of the larger modules, which lends itself well to the agile approach of developing smaller increments that ultimately add up to the final system, and not preparing a big design up front. In order to develop recommendations for the integration the authors analyze Scrum and XP to see how they are able to adopt UCD activities, specifically how they can utilize UCD requirements. The Requirement Framework introduced in the following section offers a way to approach this.

5 User Centered Requirements

Based on a generalized UCD model defined in DIN EN ISO 13407 [5], Zimmermann & Grötzbach [8] describe a Requirement Engineering framework where three types of requirements are generated, each of which constitutes the analysis and design outcome for one of the three UCD activity types. Usability Requirements are developed during the Context of Use analyses; which revolves mainly around the anticipated user, their jobs and tasks, their mental models, conceptions of the usage of the system, physical environment, organizational constraints and determinants and the like. It is important to elicit these findings from actual users in their context of use, in order to get a reliable baseline for requirements pertaining to users' effectiveness, efficiency and satisfaction. These requirements can be used as criteria for the system and intermediate prototypes through Usability Tests, questionnaires, or expert based evaluations. The Workflow Requirements focus on individual workflows and tasks to be performed by a user. Task performance models are elicited from users, the workflow is optimized for, and an improved task performance model is generated. The outcome of this module is a set of requirements pertaining to a specific user's interaction with the system in the context of a specific workflow or task, e.g. as described in use case scenarios. The requirements describe the discrete sub-steps of a user's interaction flow and the expected behavior of the system for each of these steps in an optimized workflow. It is important to validate these requirements against the usability requirements with users, e.g. by comparing an optimized workflow to the current state of workflow performance with regard to effectiveness, efficiency and user satisfaction. Workflow Requirements are ideal input for test cases, against which prototypes or the final system can be tested, either through usability tests or expert evaluations. The User Interface (UI) Requirements, generated in the Produce Design Solution activities, define properties of the intended system that are derived from Usability or User Requirements, e.g. interaction flow or screen layout. During the development phase, the UI Requirements provide guidance for technical designers regarding the information and navigation model, which can then be aligned with other technical models. They also help programmers implement the required functions using the correct visual and interaction model. UI requirements serve as criteria for the actual system that has been developed, i.e. to determine if it follows the defined model for layout and interaction. These evaluations can be user or expert based, and can be conducted during system design and testing. By translating UI Requirements into test cases, this evaluation step is facilitated.

6 Proceedings

The authors used the User Centered Requirements summarized above as a basis for a gap-analysis in order to determine whether the two agile SE Models (Scrum and XP) consider the three types of requirements adequately. As the different requirements have distinct stages, they have to be elicited, implemented and evaluated appropriately. The fulfillment of the requirements will guarantee user centeredness in the development process. In order to prepare the gap-analysis the authors used the

description of the different requirements to derive several criteria used for the assessment. The goal was to specify criteria which apply to both models. Thereby there is no 1:1 relation of the stages (elicitation, implementation, evaluation) and the criteria derived for the different types of requirements. Thus, there might be no criteria at a specific stage for a specific type of requirement, as the framework suggests. As an example, selected criteria for the UI Requirements are shown in Table 1.

Table 1. Selection of criteria, defined for the UI Requirements, based on the definition in 5.

| Elicitation | Implementation | Evaluation |
|---|--|---|
| develop appropriate representation of workflow by UI designer | verify feasibility | evaluate if UI meets UI methods to measure improvements in effectiveness and efficiency |
| specify interaction and behavioral detail | transform architecture into design solutions | verify requirements and refine existing reqs |

According to the criteria the two agile models Scrum and XP have been analyzed regarding whether the criteria's are met. This allows comprehensive statements about the considerations of UE activities and outcomes in agile SE.

The analysis results are presented by each type of requirement and in the order of the three stages and are based on the model description from the sources cited above. Subsequent to the analysis the authors give recommendations for the two agile SE Models that enhance the consideration of the three requirement types in Scrum and XP.

6.1 Implementation of User Centered Requirements

Table 2. Selection of criteria, defined for the Usability Requirements. + fulfilled; - not fulfilled; o partly fulfilled

| Usability Requirements | | SCRUM | XP |
|------------------------|---|-------|----|
| Elicitation | observe user in context of use | — | — |
| | consider workflow-oriented quality criteria | — | — |
| | measurable, verifiable and precise usability requirements | — | — |
| | gather and consider stakeholders input | o | o |
| Evaluation | verify if requirements are met | — | — |
| | measure end user's satisfaction | — | — |
| | check requirements and refine existing requirements | o | — |

The results of the analysis for the Usability Requirements (Table 2) show, that neither Scrum nor XP consider this type of requirements appropriately. During the elicitation of Usability Requirements only one criteria, the consideration of stakeholder input is partly fulfilled by both models. The insufficient acquaintance of overarching Usability Requirements can also be determined in evaluation activities. Just one criterion is met by the Scrum Models to some extend.

Table 3. Selection of criteria, defined for the Workflow Requirements.

| Workflow Requirements | | SCRUM | XP |
|-----------------------|---|-------|----|
| Elicitation | specify system behavior for given task, related to concrete goal | — | — |
| | check if new workflow is an improvement from users perspective | — | o |
| Evaluation | check correctness and completeness of workflow description | + | + |
| | check workflow for correctness, completeness and possibly find new requirements | — | — |
| | verify requirements and refine existing requirements | + | + |
| | verify that final system meets requirements | o | + |

During the elicitation of Workflow Requirements hardly any of the criteria could be found in Scrum or in XP. Except that the XP model does partly fulfill the criteria to verify if the new workflow is an improvement from the user's perspective. However, the agile models possess solid strengths in the evaluation of user requirements. The only criterion which is not met by both models is the verification of workflow mockups against the improved workflow. The impact for the usability, because of these unconsidered criteria, is negligible.

Table 4. Selection of criteria, defined for the User Interface Requirements.

| User Interface Requirements | | SCRUM | XP |
|-----------------------------|---|-------|----|
| Elicitation | develop appropriate representation of workflow by UI designer | — | — |
| | specify interaction and behavioral detail | — | — |
| Implementation | verify feasibility | + | + |
| | transform architecture into design solutions | + | + |
| Evaluation | evaluate if UI meets UI requirements | o | — |
| | concluding evaluation to see if system meets requirements | — | + |
| | methods to measure improvements in effectiveness and efficiency | — | — |
| | verify requirements and refine existing requirements | + | — |

The elicitation of User Interface Requirements is not provided by any of the two models. However, for the criteria of implementation activities, both models provide an opportunity to verify feasibility of certain interaction concepts and consider technical constraints for design decisions before the UI concepts are implemented. In terms of the evaluation of UI Requirements the two models have several distinctions. The Scrum Model provides a way to verify UI Requirements with users and experts whereas there is no information about a comparable activity in the literature for the XP Model. As opposed to that, XP does perform concluding evaluations to see if the system meets the UI Requirements within the scope of automated tests. Both models do not consider measuring the improvements of the user's effectiveness and efficiency.

6.2 Conclusion and Recommendations

Looking at the summarized results it becomes apparent that both agile models have significant deficiencies in handling User-Centered Requirements. Usability Requirements are treated insufficiently in the important stages of development. Regarding to more detailed requirements the agile models possess certain strengths and the potential for the integration with UE activities.

Workflow Requirements for instance are dealt with appropriately regarding evaluative activities. But it needs to be assured that they are elicited and processed adequately during previous stages from an UE standpoint. The development can be essentially influenced on the granular level of UI Requirements. However, the UI requirements have to be derived from correct workflow descriptions and qualitative Usability Requirements. The recommendations listed below, provide suggestions to endorse the two models in order to include the criteria of User-Centered Requirements and ensure the usability of a software product. The recommendations are derived from the results of the analysis described above.

In the descriptions both models an explicit exploration phase prior to the actual development is mentioned in which the development teams work out system architecture and technology topics to evaluate technical feasibilities, while customer stakeholders generate Product Backlog Items (in Scrum) or User Stories (in XP). Compared to common UE analysis activities the exploration phases in Scrum and XP are rather short and are supposed to not exceed one usual process iteration. Nevertheless, this exploration phases can be used by UE experts to endorse the particular development teams in a rough exploration of the real users in their natural work environment. In order to stay agile it is important to not insist on comprehensive documentation of the results, rather than emphasizing on lightweight artifacts and sharing the knowledge with the rest of the team. Having an UE domain expert in the development team also assures that generic Usability Requirements are taken into account during requirement gathering activities.

Due to the vague definition of the customer role in Scrum and XP it is not guaranteed that real users are among the group of customer stakeholders. From an UE point of view, it is essential to gather information regarding the context of use and the users' workflows and to validate early design mockups and prototypes with real users. Therefore, it is necessary to explicitly involve users on-site for certain UE activities instead of different customer stakeholders, even though when they claim to have a solid knowledge of the end users needs.

The Product Backlog (in Scrum) and the User Stories (in XP) would be the right place to capture Workflow Requirements. However, there is the risk of loosing the "big picture", of how single system features are related to each other, because both artifacts focus on the documentation of high level requirements instead of full workflows. Modeling the workflow with Essential Use Cases and scenario based descriptions [11] would be sufficient, but is not intended by any of the two models.

Scrum and the XP do not intend to perform usability tests to verify if the requirements are met, nor they measure the users satisfaction, e.g. using questionnaires. However, the Sprint Review in Scrum offers facilities to expert evaluations involving people with UE expertise and/or real users among the Scrum Team and as attendees of the Sprint Review. This can not substitute comprehensive usability evaluations, but helps to avoid user problems at an early stage.

System testing in terms of usability is a problem in agile models because the solutions are specified, conventionalized and developed in small incremental steps. However, to perform a usability test with real users, the system has to be in a certain state of complexity to evaluate the implementation of their workflows. In traditional SE models, also using incremental development, these workflows and regarding requirements are documented beforehand and a prototype could be developed regarding such a set of requirements for one workflow to be tested with the users. It certainly does not make sense to demand for Usability Testing subsequent to each process iteration, but the tests could be tied to a release plan.

Agile models provide good opportunities for a close collaboration between developers and designers during development activities. Due to the overlapping development phases and the multidisciplinary of the development teams, the feasibility of certain interaction models can be compiled with developers frequently and without fundamentally slowing down design and implementation activities. Regarding to this, design decisions can consider Usability Requirements and technical constraints in an easy and early stage. In terms of the evaluation of UI Requirements the two models differ in their proceedings. The Sprint Review in Scrum can be used to review the user interface in order to verify whether the design meets the previously defined specifications - presuming that those specifications have been created and defined as Sprint Goals beforehand. XP does not stipulate a review meeting like the Scrum Model. Unlike to Scrum, the XP Model explicitly demands for constant testing on a frequent basis. Certain subsets of UI Requirements are suited for automated test, e.g. interaction or behavior related requirements. But it is barely possible to test the conformity to a style guide regarding the accurate implementation.

7 Summary and Outlook

The underlying criteria for the assessment do not claim to be exhaustive. Anyhow, they show the right tendencies and allow to make statements in terms of the realization in the particular models. The approach presented in this paper is used to evaluate how agile software engineering (SE) models consider activities of usability engineering (UE) in order to ensure the creation of usable software products. The user-centeredness of the two agile SE models Scrum and XP has been analyzed and the question how potential gaps can be filled without losing the process agility was discussed.

As requirements play a decisive role during software development, either in software engineering but also in usability engineering, the authors assumed that requirements can serve as the common basis on which agile SE models can work

together with the results of usability engineering activities. The User Centered requirements, defined by Zimmermann and Grötzbach, describe three types of requirements derived from the results of UCD activities outlined in DIN EN ISO 13407 [5].

By using these three types of requirements the authors derived more specific criteria in order to perform a gap-analysis of the two agile models. As a result, the fulfillment of the criteria allowed comprehensive statements about the considerations of UE activities and outcomes in agile SE. It turned out that both agile models have significant deficiencies in handling User-Centered Requirements. Usability Requirements are treated insufficiently in all the important stages of development.

The presented approach has been used to acquire first insights about the ability of agile SE models in creating usable software. However, the authors are well aware of the need for further more extensive and more specific criteria. Using and applying them to other agile models will enable to derive more generic statements about the integration of UE in agile SE models in general.

References

- [1] Beck, K.. Extreme Programming explained, Addison-Wesley, Boston, (2000)
- [2] Boehm, B.. A Spiral Model of Software Development and Enhancement. IEEE Computer. Vol. 21. pp. 61-72, (1988)
- [3] Cohn M.: User Stories Applied – For Agile Software Development, Addison-Wesley:Boston, (2004)
- [4] Cooper, A.: *About Face 2.0*. Wiley Publishing Inc., Indianapolis, (2003)
- [5] DIN EN ISO 13407: Human-centered design processes for interactive systems. Brussels, CEN - European Committee for Standardization, (1999)
- [6] DIN EN ISO 9241-11. Ergonomic requirements for office work with visual display terminals (VDTs) – Part 11: Guidance on usability. International Organization for Standardization, (1998)
- [7] Ferre, X. Integration of Usability Techniques into the Software Development Process. Proceedings of the 2003 International Conference on Software Engineering. pp. 28-35. Portland, (2003)
- [8] Groetzbach L., Zimmermann D.: A Requirement Engineering Approach to User Centered Design. In preparation, HCII 2007, Beijing, (2007.)
- [9] Larman C.: Agile & Iterative Development – A Manager’s Guide, Addison-Wesley: Boston. (2004)
- [10] Mayhew, D.J.: The Usability Engineering Lifecycle. Morgan Kaufmann, San Francisco (1999)
- [11] Rosson M. B., Carrol J. M.: Usability Engineering: Scenario-Based Development of Human-Computer Interaction, Academic Press, London, (2002)
- [12] Royce, W.. Managing the Development of Large Software Systems. Proceedings of IEEE WESCON. Vol 26. no. August. pp. 1-9. (1970)
- [13] Schwaber, K. & M. Beedle. Agile Software Development with Scrum. Prentice Hall: Upper Saddle River, (2002)