Continuous Requirements Engineering and Human-Centered Agile Software Development

Peter Forbrig

University of Rostock, Chair in Software Engineering,
Albert-Einstein-Str. 21,
18051 Rostock, Germany,
Peter.Forbrig@uni-rostock.de

Abstract. The idea of Continuous Requirements Engineering in relation to a Human-Centered Agile Development Process is discussed. First, it is argued that Continuous Requirements Engineering has to cover design-time and runtime aspects. In this way maintenance is covered as well. Second, arguments are provided for integrating aspects of usability and user experience into requirements specifications. This has to be done continuously. Therefore, the term Continuous Human-Centered Development is introduced and discussed.
Based on a process model for SCRUM some aspects of integrating HCD into the development process are discussed.

Keywords: Specification, Continuous Human-Centered Design, Process Modeling, Requirements Engineering, Requirements Models, Agile Development, SCRUM.

1 Introduction

Agile software development methods like Scrum have become popular during the last years because of their flexibility to adapt to dynamic changing application domains and changing user's needs during software development. They prevent the failure of projects because otherwise it takes too much time from finalized requirements specifications to first tests of the developed system However, these agile methods still focus on a limited period of time for the software development. A longer period of the application of the developed software and their maintenance are not part of these methods.

However, monitoring of running systems might be useful. In this way Continuous Requirements Engineering might be a way to focus on the integration of software development and maintenance for agile development methods.

Additionally, we believe that especially a human-centered approach leads to software systems that are usable and provide good user experience. Its integration into agile methods is another challenge.

The paper is structured in the following way. First, we describe our point of view of Continuous Requirements Engineering and its further development. Second, the Human-Centered Development Process is discussed. Third, an agile development process is suggested that includes Continues Requirements Engineering and as a part of the Continuous Human-Centered Development. The paper closes with a summary and an outlook.

2. Continuous Requirements Engineering

Current engineering-based approaches are rooted into well elaborated systems models, enterprise architectures, ontologies, and information logistics representations. They provide transparency, reliability, and security in the whole lifecycle of the system. Currently such approaches are designed and mainly applied to large enterprises that have relatively long change cycles. In case such changes have to be performed more frequently a much higher flexibility is required. For such systems the engineering processes grow into continuous engineering that requires continuous requirements engineering (CRE). CRE can only be successful if it combines rigid engineering principles with agility, emergence, and spontaneity to support sustainability and viability of the systems under development.

Smaller scale enterprises need new approaches, methods, and tools to be capable to embrace the growing variety of opportunities and challenges offered by fast changing and hardly predictable environment. In this type of systems, continuous requirements engineering can be a solution if it is integrated with management and design approaches.

It is well known that flawed requirements cause a lot of problems. Some projects totally fail because of that, others waste a lot of money because the correction of resulted errors in the implementation is very time consuming and labor intensive. Therefore, new ideas in identifying the correct requirements are very important.

2.1. Related Work

A framework for Continuous Requirements Engineering for self-adapting systems was provided by Qureshi et al. in [20]. The domain of self-adapting systems was selected because many software systems like service bases systems are running continuously in an open environment like the Internet. “The only way to understand what changes are acceptable in a system is with respect to its requirements, and more specifically, its intentions.” 19Users are allowed to provide requirements during runtime in terms of models that are goal- and user-oriented. These models lead to adaptations of a system.

They provided a framework that is called CARE (Continuous Adaptive Requirements Engineering) for building self-adapting systems. They distinguish requirements engineering during design-time and run-time that is related to design-time reasoning and run-time reasoning for such specific systems.

Leah Goldin et al. [13] discuss the question whether in the development of large scale systems the institutionalized, proactive requirements reuse pays off. In their case study they found out that at least for the studied project it paid off to meet the moving target of requirements based on existing specifications.

Reuse of requirements specification might be one way to reduce time to market. However, there are still a lot of aspects to consider like the kind of specification languages for functional and non-function requirements.

For the handling of BPMN and S-BPM specifications concepts for reusable components were presented in [9] and [10]9. Following this approach with appropriate tool support would very much help to quickly update requirements specifications.

Workflow management systems can support the execution of business process specifications. Fleischmann et al. [5] follow this argumentation line by using S-BPM: ”When agile project structures and active involvement of concerned stakeholders become part of organizational change, requirements to software development might change continuously. Hence, the effort for transforming representations from requirements specification to executable design models should be minimized.”

The transformation can be omitted if requirements can be interpreted directly or if the transformation process can be automated. We will come back to these aspects in the following paragraph.

2.2. Discussion of CRE

The idea of Continuous Requirements Engineering allows the adaptation of requirements not only during the whole design time but also during runtime. Even the adaptation of requirements during the whole design time is already an innovation for a lot of current projects. To extend this option of changing requirements to runtime is even a bigger challenge.

Within the S-BPM approach the executable design is directly specified in a notation of a diagram. These specifications are interpreted during runtime. Therefore, no transformation is necessary. Users can articulate their requirements by providing refined specifications of behavior components. This manipulation of S-BPM diagrams changes the control flow of the running system. There is even no need to stop the running system.

Indeed, the approach is very similar to CARE presented by Qureshi et al. in [20]. The only difference is the different kinds of models that have to be manipulated and provided. S-BPM asks for behavior specifications while CARE uses requirement models that are goal- and user-oriented. They are extended goal models with the notion of user attitudes that are specified as preferences between model elements. Optional requirements are possible as well.

In both cases, S-BPM and CARE, users are considered to be as possible creators of such models. However, the manipulation of models can also be done by any stakeholder.

Additionally, in cases where models can drive the generation process, as in the MDA approach, requirements models can be transformed automatically to runtime models for execution. We will come back to this later in the context of Human-Centered Design.

3. Human-Centered Design

In the same way as agile development methods are popular for software engineering experts Human-Centered Design (HCD) is popular for usability and user experience experts. It focusses on tasks users have to perform, usability and user experience, aspects that do not play their important role for software engineers in general. They focus often on the technical aspects of an application only.

One of the main reasons for the success of HCD is that the context of use and the evaluation of design solutions play an important role. User requirements are more important than technical features that software engineers might like. It is more likely that users get what they really want.

3.1. Related Work

The HCD process is standardized by ISO 9241-210. It consists of a planning phase and four phases that are performed in an iterative way.

Within the first phase analysts try to understand the context of use. Stakeholders are identified. Their roles and tasks are analyzed and typical application scenarios are specified. Additionally, artefacts and tools they work with are captured. Last but not least the environment in which the application has to be performed is analyzed. This is done according to the location, the surrounding objects and people. Sometimes the available services are important as well.

Based on this analysis user requirements are specified. Additionally to the goals of users functional and nonfunctional requirements are collected. Domain specific requirements might be important as well. This is e.g. the case when domain specific standards exists that have to be fulfilled by the application.

First design solutions are produced afterwards to fulfill the identified requirements. Such design solutions include first ideas of user interfaces.

The design solutions are evaluated in the last phase of the HCD process. If the requirements of the users are met the development process comes to an end and the implementation of the application core can be performed. Otherwise, there are three possible continuations. If there are serious problems one has to analyze the context of use again and has to proceed with the first phase. In case the general analysis of the context of use seems to be correct but some requirements were specified in the wrong way, one has to rewrite some requirements or identify some new ones. Finally, it can be possible that only new design solutions are necessary. In this case requirements are specified in the right way but the design solutions have to be improved. **Fig. 1**gives a visual impression of the discussed HCD process model**.**

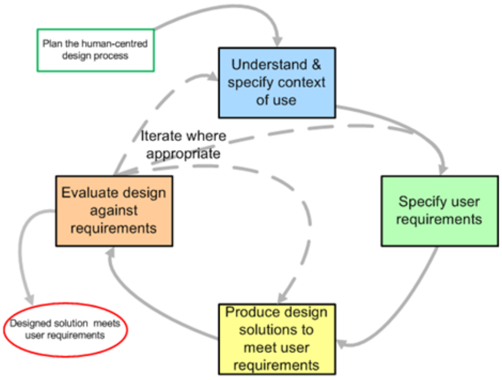
Plan the human-centred design process
Understand & specify context of use
Iterate where appropriate
Specify user requirements
Evaluate design against requirements
Produce design solutions to meet user requirements
Designed solution meets user requirements

Fig. 1. The design process from ISO 9241-210– Human-centered design process (from https: //thestandardinteractiondesignprocess.wordpress.com/).

Fig. 1 provides a good overview of the main ideas of the HCD process. Unfortunately, the process model does not consider the integration of HCD into an agile development process. Paelke et al. [17] published a process model and called it Agile UCD-Process. (User-Centered Design was the predecessor of HCD.). It is visualized in the following figure.

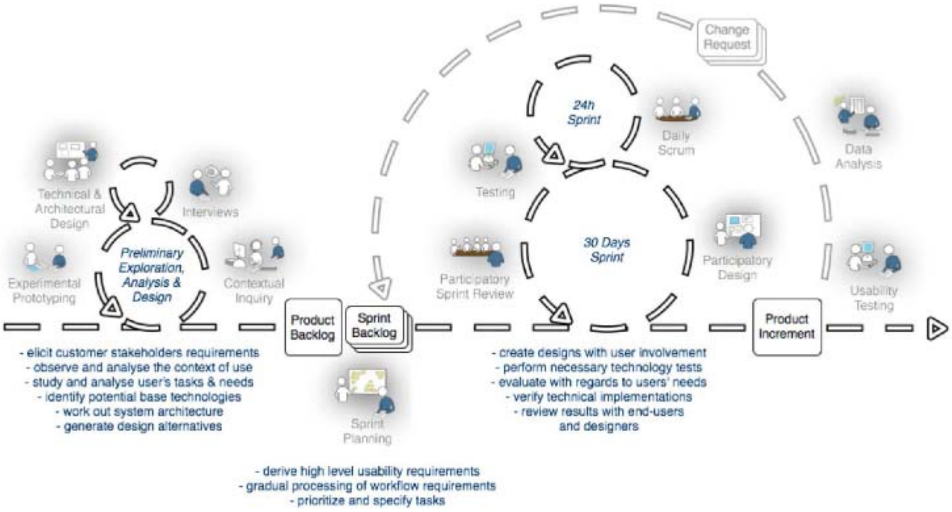
Change Request
24h Sprint
Daily Scrum
Data Analysis
Testing
Technical & Architectural Design
Interviews
Preliminary Exploration, Analysis & Design
Experimental Prototyping
Contextual Inquiry
30 Days Sprint
Participatory Sprint Review
Participatory Design
Usability Testing
Product Backlog
Sprint Backlog
Product Increment
Sprint Planning
- elicit customer stakeholders requirements
- observe and analyse the context of use
- study and analyse user's tasks & needs
- identify potential base technologies
- work out system architecture
- generate design alternatives
- create designs with user involvement
- perform necessary technology tests
- evaluate with regards to users' needs
- verify technical implementations
- review results with end-users and designers
- derive high level usability requirements
- gradual processing of workflow requirements
- prioritize and specify tasks

Fig. 2. Agile User-Centered Design Process (from Paelke et al. [17]).

The presented process model suggests having a common initial phase for developers and HCI specialists. Afterwards there are activities of both groups. Unfortunately, it is not quite clear in which order these activities are performed. Additionally, the requirements elicitation is a little bit too much uncoupled from the software development process. A stronger coupling was suggested by Paul et al. [18] and is presented by the following Figure. Additionally, it provides the names of models like user or task model that have to be specified in the corresponding state of the software development

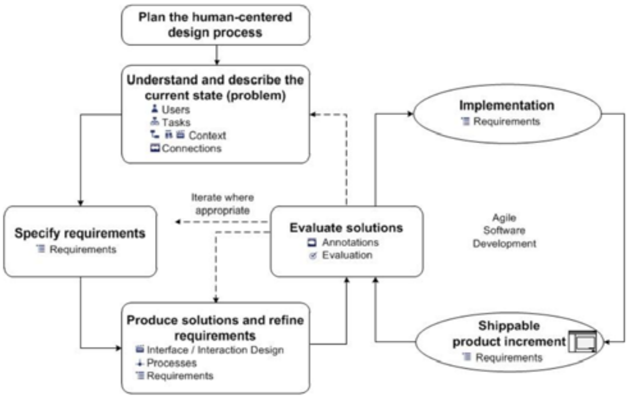
Plan the human-centered design process
Understand and describe the current state (problem)
Users
Tasks
Context
Connections
Implementation
Requirements
Iterate where appropriate
Specify requirements
Requirements
Evaluate solutions
Annotations
Evaluation
Agile Software Development
Produce solutions and refine requirements
Interface / Interaction Design
Processes
Requirements
Shippable product increment
Requirements

Fig. 3. Extended User-Centered Design Process (Paul et al. 17).

There are several attempts to integrate HCI aspects like usability into agile development processes. Examples of discussions about process models can also be found in [21] and [23].

Sy 24 suggest two interleaving processes for developers and HCI specialists, which are called interaction designer in her terminology. She suggests that at the beginning there has to be a common plan and some user data have to be gathered. Afterwards, developers start in the first development cycle with implementations that are not much related to the user interface. This could be e.g. certain services the application is based on with simple user interfaces.

In parallel HCI specialists provide certain design solutions for cycle two and gather customer data for cycle three.

In cycle two developers implement the design solutions from cycle one and in parallel their code from cycle one is tested by HCI experts. Additionally, they design for the next cycle and analyze for the cycle after the next cycle. This is the general development pattern. In some way interaction designers work two cycles ahead to developers in analyzing customer data and one cycle ahead in developing design solutions. A similar approach by separating the activities of analysts and developers was presented in [11] for the SCRUM approach.

3.2. Continuous HCD

Human-Centered Development should also not be done only once during design time. It has to be continuously performed during the whole development process. It has even to be processed during runtime.

Fig. 4 presents an updated version of the suggested development process. It starts with an initial phase where all participants in the development process agree on a vision and provide needs that have to be supported and fulfilled by the software under development.

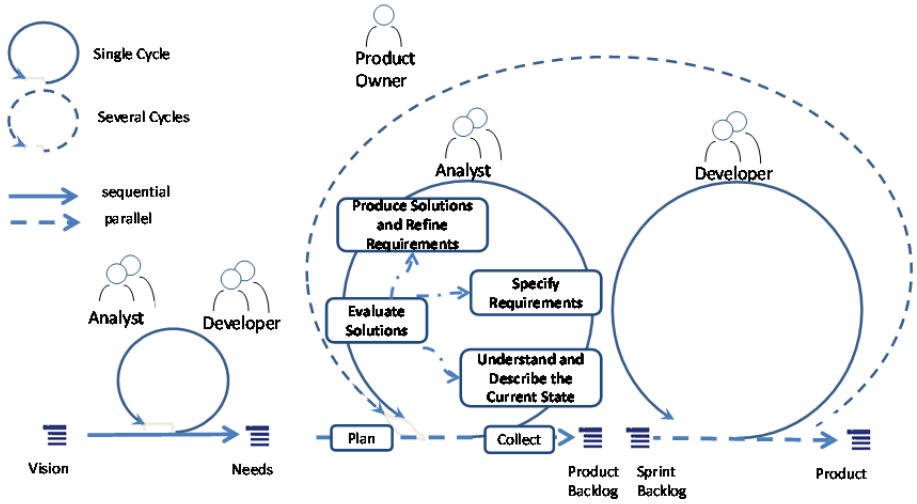

Fig. 4. Human-Centered Design Process for SCRUM

The development cycle of analysts (within the cyclic iteration) is executed in parallel to the cycle of the developers. Details of the necessary specifications within this cycle are presented in **Fig. 5**. It should run at least one cycle ahead. However, some companies reported privately that they successfully applied the characterized approach without analyzing the human aspects one cycle ahead. In such cases user interface evaluation did not yield to new requirements. In general, it would be better to analyze more precisely and evaluate different alternative solutions.

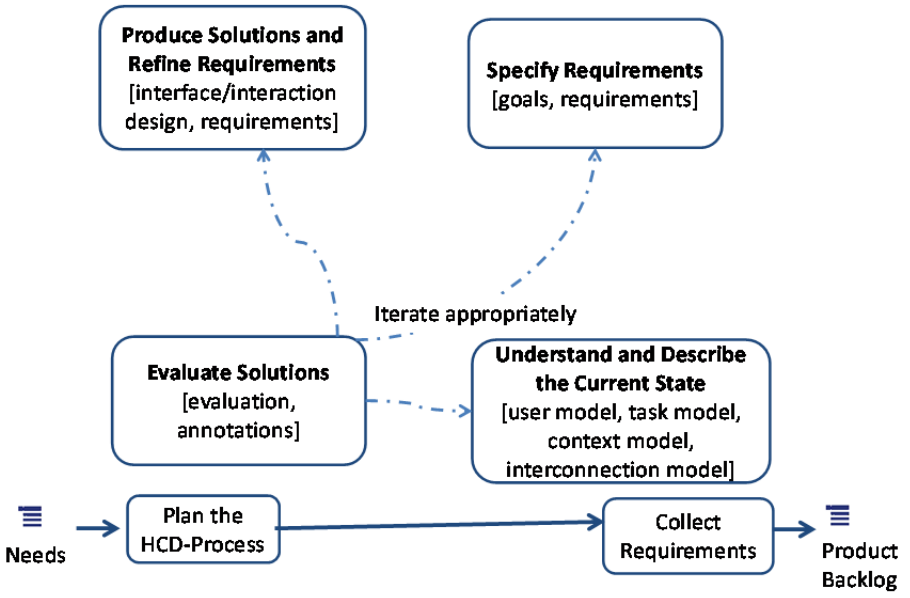
Produce Solutions and Refine Requirements [interface/interaction design, requirements]
Specify Requirements [goals, requirements]
Iterate appropriately
Evaluate Solutions [evaluation, annotations]
Understand and Describe the Current State [user model, task model, context model, interconnection model]
Needs
Plan the HCD-Process
Collect Requirements
Product Backlog

Fig. 5. Detailed Human-Centered Design Process for Analysts (from our paper [11]).

Fig.5 provides some details of the HCD process that should run in parallel to the software development process. Such details are specific models that should be available und supported by tools like user model, task model or interconnection model. Paul 18 provides a usability repository for all these models. The tool was evaluated in an industrial context and found to be helpful.

This may be related to the fact that the human aspect becomes more and more important for interactive systems. Usability and user experience are key factors of success or failure of software.

It is important to have UX-specialists within the development team. Additionally, all members of the development team should be trained in the fundamentals requirements elicitation and usability evaluation. There has to be a common ground on these aspects.

Kuusinen analyses in 13 the allocation of tasks between HCI specialists and developers in agile development projects. Her studies delivered two main results.

- First, HCI specialists and developers cooperate on user-interface design, while other UX aspects are downplayed.
- Second, many UX-related tasks were successfully handled by developers alone.

Kuusinen suggests a task-oriented integration approach especially for projects with minimal UX resources. This is in line with the suggested process model, where task models have to be created at the beginning of the HCD process.

Innovation has always to be discussed from a human perspective. Technological innovation is important but it has to consider humans in their role as different stakeholders.

Agile development methods often focus on customers while User-Centered Design focusses on users. Both aspects have to be considered during Continuous Human-Centered Design.

We already mentioned the possibility of generating software based on models. We have been working for several years on task-based generation of interactive systems. Fischer et al. [7] provide a model-driven approach for user-interface design. This might allow improving the opportunity to evaluate different design solutions because no programming is necessary. Domain knowledge is specified as model. In case of user interfaces these models are task models, user models, application domain model, platform model, and environment model. Based on these models an abstract user interface (AUI) is constructed that later is refined to a concrete user interface (CUI) and at the end to final user interface. The abstract user interface should be independent of the of the destination platform. CUIs are already platform specific. Final user interfaces consider e.g. already the size and position of objects and their colors. This idea is known as the CAMELEON [4] approach. Its application to model-driven user-interface generation is presented in the following figure. (M2M means model-to-model transformation and M2C model-to-code.)

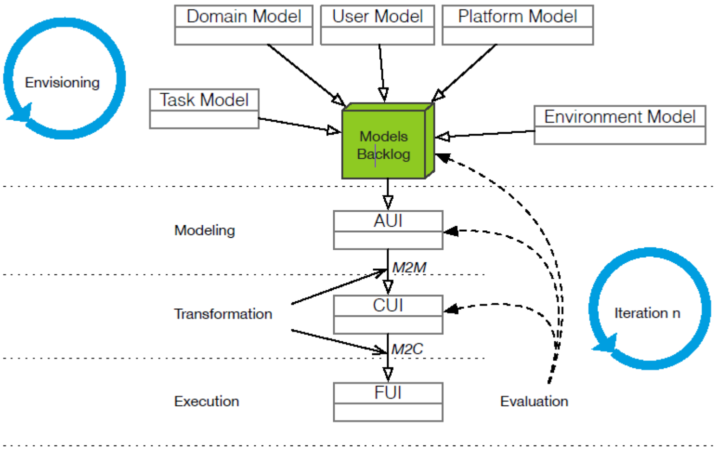

Fig. 6. Model flow and evaluation feedback (from [7]).

Supporting user interface generation based on models by tools allows the exploration of different designs alternatives in a cheap way. The production of solutions in the HCD process is supported effectively in this way. There might be a new push for applying models if the model-driven development of user interfaces can be used within the agile context.

4. Summary and Outlook

The paper discussed the ideas of Continuous Requirements Engineering and Human-Centered Agile Software Development. It argues for having Continuous Requirements Engineering during the whole design-time as well as during the whole runtime. In this way changing requirements have be considered all the time and projects have to be managed during the whole life-cycle of a software system. Development teams can be reduced in size. However, according to this approach they should be always available. The analysis phase never stops.

Continuous Human-Centered Development is suggested as part of the Continuous Requirements Engineering, which means an integration of aspects of usability engineering as well as requirements engineering into the continuous agile development process. A process model for integrating Continuous Human-Centered Development into Continuous Requirements Engineering is provided and possible applications of model-driven technologies for UI-aspects are discussed.

References

1. Agile Manifesto, http://agilemanifesto.org/, last visited June 4th 2015.
2. Baresi, L., and Ghezzi, C. C.: The disappearing boundary between development-time and run-time. In Future of Software Engineering Research, 2010.
3. Claes, J. , Vanderfeesten, I. Reijers, H. A., Pinggera, J., Weidlich, M., Zugal, St., Fahland, D., Weber, B., Mendling, J., and Poels, G.: "Tying process model quality to the modeling process: the impact of structuring, movement, and speed," in Business Process Management, Springer, Berlin, pp. 33-48, 2012. http://dx.doi.org/10.1007/978-3-642-32885-5_3
4. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J. A Unifying Reference Framework for Multi-target User Interfaces. In Interacting with Computers, Vol. 15. Elsevier, 2003. pp. 289-308.
5. Fleischmann, A., Schmidt, W., and Stary, C.: Requirements Specification as Executable Software Design – A Behavior Perspective, in 14 pp. 9-18, 2015.
6. Fleischmann, A., Schmidt, W., and Stary, C.: Dynamic Socio-technical System Design based on Stakeholder Interaction, Complex Systems Informatics and Modeling Quarterly, No. 3, 2015, pp. 63-83. https://csimq-journals.rtu.lv/article/view/csimq.2015-3.04
7. Fischer, H., Yigitbas, E. and Sauer, S.: Model flow and evaluation feedback, In: Beckmann Ch. and Gross T. (Eds) INTERACT 2015 Adjunct Proceedings, pp. 201 -208.
8. Fitzgerald, B. and Stol, K.-J.: Continous software engineering and beyond: trends and challenges. In Proc. 1st International Workshop on Rapid Continuous Software Engineering – RcoSE 2014, ACM, New York, NY, USA, pp. 1-9.
9. Forbrig, P.: Generic Components for BPMN Specifications Perspectives in Business Informatics Research - 13th International Conference, BIR 2014, Lund, Sweden, September 22-24, 2014. Proceedings , Seite 202--216.2014DOI: 10.1007/978-3-319-11370-8_15
10. Forbrig, P.: Reuse of models in S-BPM process specifications, Proceedings of the 7th International Conference on Subject-Oriented Business Process Management, S-BPM ONE 2015, Kiel, Germany, April 23-24, 2015 , Seite 6-16. 2015, DOI: 10.1145/2723839.2723846
11. Forbrig, P. and Herczeg M.: Managing the Agile Process of Human-Centred Design and Software Development, In: Beckmann Ch. and Gross T. (Eds) INTERACT 2015 Adjunct Proceedings, pp. 223 -232.
12. Frishammar, J., Lichtenthaler, U., Richtnér, A. (2013). Managing process development: key issues and dimensions in the front end. R&D Management, 43(3), 213-226.
13. Goldin, L.,and Berry, D. M.: Reuse of requirements reduced time to market at one industrial shop: a case study, Requirements Engineering, Springer, vol. 20, Issue 1, pp. 23-44, 2015. http://dx.doi.org/10.1007/s00766-013-0182-7
14. Kuusinen, K.: Task allocation between UX Specialists and Developers in Agile Software Development Projects, In: J. Abascal et al. (Eds.): INTERACT 2015, Part III, LNCS 9298, pp. 27–44, 2015. DOI: 10.1007/978-3-319-22698-9_3
15. Matulevičius, R. et al. (Eds.): REFSQ Workshop proceedings, http://ceur-ws.org/Vol-1342/, 2015.
16. Memmel, T., Gundelsweiler, F. and Reiterer, H. 2007. Agile human-centered software engineering. In Proceedings of the 21st British HCI Group Annual Conference on People and Computers: HCI...but not as we know it - Volume 1 (BCS-HCI '07), Vol. 1. British Computer Society, Swinton, UK, UK, pp. 167-175.
17. Paelke, V. and Nebe, K. Integrating Agile Methods for Mixed Reality Design Space Exploration. In Proceedings of the 7th ACM conference on Designing interactive systems (DIS '08). ACM, New York, NY, USA, pp. 240-249. http://doi.acm.org/10.1145/1394445.1394471
18. Paul, M., Roenspieß A., Mentler T., Herczeg M. (2014). The Usability Engineering Repository (UsER). In Hasselbring, W & Ehmke, N C (Eds.) Software Engineering 2014 - Fach-

tagung des GI-Fachbereichs Softwaretechnik, 25.-28. Februar 2014, Kiel. Gesellschaft für Informatik e.V. (GI). pp.. 113-118

19. Paul, M. Systemgestützte Integration des Usability-Engineerings in den Software-Entwicklungsprozess, PhD Thesis, Univerity of Lübeck, 2015.
20. Qureshi, N. A., Perini, A., Ernst, N.A., and Mylopoulos, J: "Towards a Continuous Requirements Engineering Framework for Self-Adaptive Systems", In First International Workshop on RE @ Runtime at 18th IEEE International Requirements Engineering Conference (RE '10), Sydney, pp .9-16 ,Sydney, September 2010.
21. Salah, D., Paige, R. and Cairns, P. A Practitioner Perspective on Integrating Agile and User Centred Design, Proceedings of the 28th International BCS Human Computer Interaction Conference (HCI 2014),
22. Singh, M. U-SCRUM: An agile methodology for promoting usability, Integrating usability engineering and agile software development: A literature review. In Proc. AGILE 2009, IEEE Press, pp. 555-560.
23. Sohaib, O. and Khan, K.: Integrating usability engineering and agile software development: A literature review. In Proc. International Conference on Computer design and Applications (ICCDA), Volume 2, pp. 32 – 38, 2010.
24. Sutherland, J. Harrison, N. and Riddle, J. Teams That Finish Early Accelerate Faster: A Pattern Language for High Performing Scrum Teams. In Proc. HICSS 2014, pp. 4722-4728.
25. Sy, D.: Adapting usability investigations for agile user-centered design. J. Usability Stud. 2 (3), pp. 112–132, 2007.
26. Weber, H., and Mueller, H. (Eds.): Continuous Engineering for Industrial Scale Software Systems, Dagstuhl Seminar 98092, 1998.