Integrating Human-Centered and Model-Driven Methods in Agile UI Development

Holger Fischer, Enes Yigitbas, Stefan Sauer
s-lab – Software Quality Lab
University of Paderborn
Zukunftsmeile 1, 33102 Paderborn, Germany
{hfischer, eyigitbas, sauer}@s-lab.upb.de

Abstract. The development of interactive systems is a challenging task and requires structured methods and processes for developing high quality user interfaces. Existing approaches for developing interactive systems are mainly based on the model-driven user interface development (MDUID) and human-centered design (HCD) paradigms. Although these approaches support the efficient development of interactive systems considering human factors, they lack in flexibility to cover dynamic characteristics in the development process. To overcome this deficit we propose an agile UI development process that combines strengths of MDUID and HCD and at the same time enables an incremental development process having a continuously runnable part of the software front-end ready for reviews with the end user.

1 Motivation

Today, the usability of an interactive system has been recognized as an important quality aspect within software development industry. The development of the interaction between humans and interactive systems is a challenging task and requires structured methods and processes for developing and maintaining high quality user interfaces.

Model-driven development (MDD) [1] is a current development paradigm that addresses the creation of domain-specific and formalized models with the aim of generating a code base. Abstraction is a key factor of MDD to describe domains regardless of the target programming language or platform. Hence, problem descriptions can be specified with more precision and free from redundancies. Furthermore, the traceability of changes and the reusability of models are supported using specific tool chains for transformation and generation. However, MDD only focuses on the users in the initial analysis. There is no continuous participation of users during the development stages. A formative or summative evaluation of the specified models or the user interface is not in focus.

Human-centered design (HCD) [2] addresses such continuous user participation during design and development. User requirements and design

solutions are refined over the time using an iterative design as well as an early focus on the users with their needs, goals and tasks. Integrating users during the whole development enables a continuous feedback and validation instead of interpretation and assumptions. Hence, obtained insights of user requirements analysis and usability tests are usually specified in a narrative way using scenarios, storyboards or reports instead of formalized models. Furthermore, HCD focuses on the development of design solutions, but not on the incremental development of UI components and the underlying system.

Agile development (e.g. Scrum [3]) enables an incremental development of software having a continuously runnable part of the software ready for reviews with the customer. Therefore, transparency and inspection are two key factors to create a common understanding within a project. However, agile development is not targeted on modeling user requirements and human behavior in a comprehensive way. Having inspections with the customer does not fulfill the expectations of having a broad analysis and evaluation with multiple users who are currently working with a system. Additionally, a systematic and documented way of decisions concerning the interaction or the UI as well as a systematic treatment of user feedback is missing.

The existing gap between human-computer interaction (HCI) and software engineering (SE) is still a challenge as well as a chance to define a systematic method for software development. The challenge addressed in this paper is to create a model-driven method for agile UI development that combines the advantages of all three paradigms mentioned above: Systematic and sustainable model-driven development, early and continuous user participation as well as incremental deployment. Thus, the open issues in every single paradigm will be fixed within their combination.

The paper is structured as following: First, we introduce some related work on the topics of model-driven HCI development and agile model-driven development. Thereafter, we present the concept of our agile model-driven UI approach. Finally, we conclude our paper and give an outlook for future work.

2 Related Work

Focusing on the topic of agile UI development integrating human-centered and model-driven methods, multiple aspects have to be taken into account. In the following we will briefly sum up existing approaches on model-driven UI development and human-centered design as well as agility in model-driven development approaches.

Model-driven User Interface Development (MDUID) brings together two subareas of software development, which are MDD and user interface development (UID). The core idea behind MDUID is to automatize the development process of UI development by making the models the primary artifact in the development process rather than application code. An MDUID process usually involves multiple UI models on different levels of abstractions that are stepwise transformed to the final user interfaces by model transformations. The CAMELEON Reference Framework (CRF) [4] provides a unified reference framework for MDUID differentiating between the abstraction levels task & concept, abstract user interface (AUI), concrete user interface (CUI) and final user interface (FUI).

There are various state-of-the-art modeling languages for covering the different abstraction levels of the CRF. For example MARIA XML (Model-based lAnguage foR Interactive Applications) [5] and IFML (Interaction Flow Modeling Language) [6] provide both an AUI modeling language and a tool-support to create and edit AUI models. Based on these AUI models further transformations can be performed to transform them into platform-specific CUI models, which eventually are needed for generating the final user interfaces (FUI). The described MDUID approaches enable the specification and also support the generation of UIs, but they do not target aspects of HCD. Therefore recommended activities of the HCD process under the terms of ISO 9241-210 are not sufficiently integrated in the MDUID approach. To overcome this deficit a model-based HCI development process was proposed by Petrasch [7]. Here the author proposes a HCD process that is based on formalized models to cover the different aspects of the HCD process like describing the context of use or specifying user requirements. Although this approach tries to align model-based UI development methods with HCD aspects, it is not focusing on the topic of agile UI development.

In order to combine MDUID and HCD in an agile manner it is important to analyze existing approaches on agile model-driven development. In this context our proposed method (see section 3) is based on the agile model-driven development (AMDD) approach by Ambler [8]. As the name implies, AMDD is the agile version of MDD. MDD is an approach to software development where extensive models are created before source code is written. The difference with AMDD is that instead of creating extensive models before writing source code you instead create agile models, “which are just barely good enough that drive your overall development efforts” [8]. The AMDD lifecycle starts with an envisioning phase where initial requirements are specified and an initial architecture envisioning is sketched. After this first iteration the actual development iterations start. Development iterations consist of a model storming,

modeling iteration and test-driven development phase. An iteration phase can be guided by optional reviews of the developed increment. The development iterations are repeated until a software product results that conforms to the user's requirements.

3 Integrated Development Method

Taking model-based HCI development and agile model-driven development into account, we propose an integrated approach (Figure 1) with a specific model flow and evaluation feedback (Figure 2) to combine

- a systematic and sustainable model-driven development,
- early and continuous user participation and
- incremental deployment.

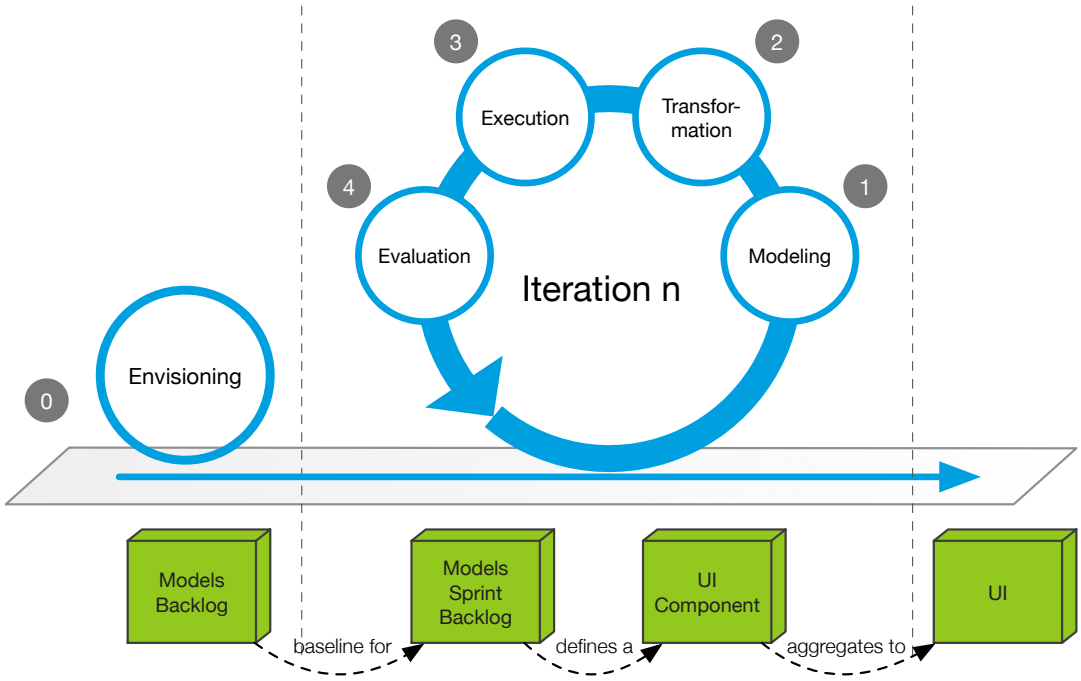

Figure 1. Agile model-driven UI development lifecycle

This approach consists of four stages and an additional envisioning up-front stage, which are described as follows:

0) Envisioning

In the envisioning up-front stage, a business analyst analyzes the context of use considering the users with their characteristics, needs, intentions, tasks, physical and social environment as well as the business goals of the organization. Then, all gathered information are described in specific models, e.g. user model, task model, domain model, platform model, environment model. These models build

up the models backlog corresponding a product backlog in agile development. This stage serves as a baseline for the development and will be extended in the modeling step during iterations.

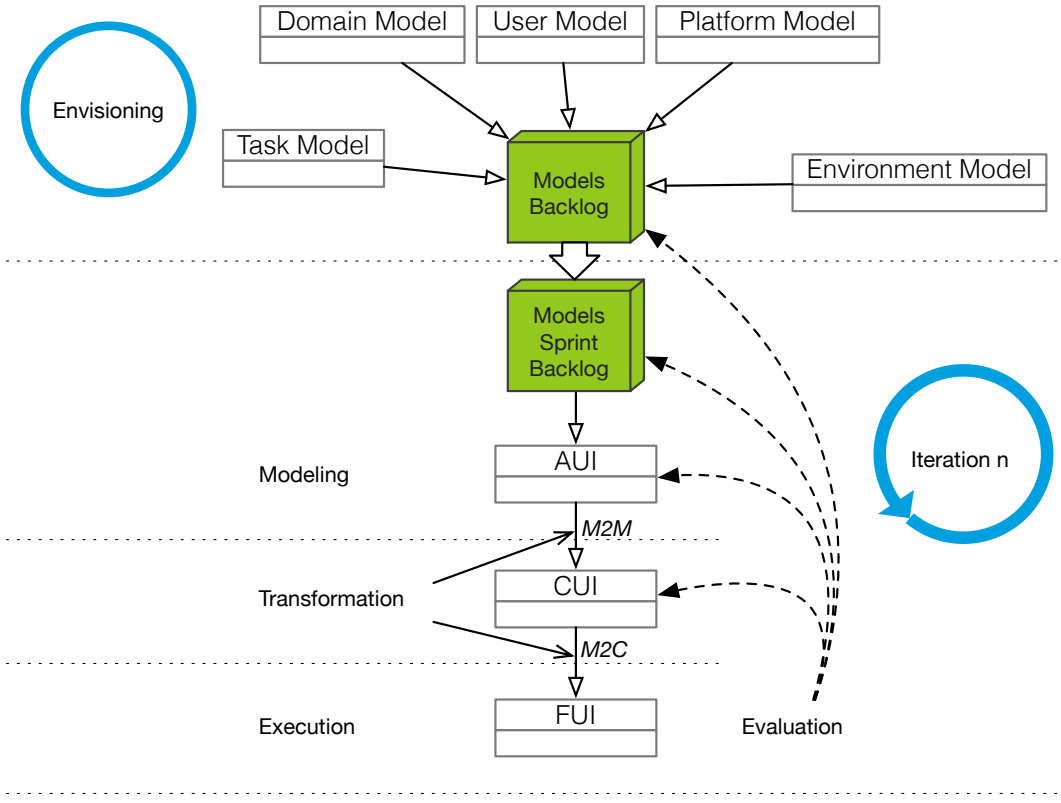

Figure 2. Model flow and evaluation feedback

1) Modeling

Each iteration (or sprint) starts with a modeling stage. Similar to a sprint backlog, the relevant models out of the models backlog will be expanded, specified as well as refined in more detail as an AUI model. This is conducted by using modeling languages like the interaction flow modeling language (IFML) [6] or MARIA XML [5] and may be enriched with formalized UI patterns like GUI patterns (e.g. wizard or auto-completion pattern) [9].

2) Transformation

Using model-to-model (M2M) transformations, the AUI model is than transformed in a CUI model specified for the target platform. Based on the CUI model the initial code for the FUI can be generated with a model-to-code (M2C) transformation.

3) Execution

Afterwards, a generated code base will be implemented with the functionality needed and which will be able to enrich with business logic. The previously implemented software component will be deployed and executed.

4) Evaluation

During the execution of a software component the behavior of the different test users will be logged in a pseudonymized way using appropriate logging mechanisms. The logged data include e.g. interaction flow, time needed to complete a task, used interaction objects or the frequency of performed tasks. These information will be analyzed e.g. to identify shortcuts in navigation paths (elephant paths), to restructure the navigation based on often performed tasks or to identify interaction barriers out of time deviations. In addition, usability tests may be carried out to gather more qualitative feedback. The model backlog will be adapted with the gathered insights of the evaluation in order to start the next iteration. According to the kind of feedback the results will be iterated on the appropriate abstraction level of the underlying models. If there are minor changes concerning the representation on the platform level the CUI models will be enhanced. Otherwise, conceptual changes have to be revised in AUI models or even further in the task and concept models.

4 Conclusion and Outlook

In this paper, we presented a concept of an agile user interface development approach enriched with human-centered and model-driven methods. Thus, we try to combine the advantages of all three paradigms in one approach: An increased usability through HCD methods and user participation; an improved sustainability using model-driven methods; continuously runnable software parts for a better communication within the development team and towards customers and users.

We believe that this combination is promising to software developing enterprises rapidly being able to create runnable and usable software releases. In this way, we focus on aspects to change software engineering processes to support usability as well as to synchronize HCD with software development activities based on fast sprints.

In our future work we will further expand the concepts in more detail. In doing so, we are thinking about logging mechanisms to study the users' behavior. Gathered data could then be abstracted within formalized models. These models could be compared with the already developed models in the backlog in order to emphasize and improve the differences between these models.

In addition, we will evaluate the concept within small software projects as well as within our teaching activities.

References

[1] Stahl, T. and Voelter, M. Model-Driven Software Development: Technology, Engineering, Management. John Wiley & Sons Ltd., Chichester, UK, 2006. ISBN 978-0470025700

[2] ISO 9241-210:2010 Ergonomics of human-system interaction – Part 210: Human-centered design for interactive systems, 2010.

[3] Schwaber, K. and Sutherland, J. The Scrum Guide, 2013. http://www.scrumguides.org/ Last visit: 05/10/15.

[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] Paternò, F., Santoro, C. and Spano, L.D. MARIA: A universal, declarative, multiple abstraction-level language for service-oriented applications in ubiquitous environments. In ACM Transactions on Computer-Human Interaction, Vol. 16 (4), Article 19. ACM, New York, 2009.

[6] Interaction Flow Modeling Language (IFML), Specification 1.0. OMG, 2015. http://www.omg.org/spec/IFML/ (Retrieved June 11, 2015)

[7] Petrasch, R. Model based User Interface Design: Model Driven Architecture and HCI Patterns. In Softwaretechnik-Trends 27 (3), 2007.

[8] Ambler, S.W. Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development. http://www.agilemodeling.com/essays/amdd.htm (Retrieved June 11, 2015)

[9] Yigitbas, E., Mohrmann, B. and Sauer, S. Model-driven UI Development integrating HCI Patterns. In Proceedings of the 7th ACM SIGCHI Symposium on Engineering Interactive Computing Systems (EICS), International Workshop on Large-scale and model-based Interactive Systems: Approaches and Challenges. 2015 (to be published)