Towards a Task Driven Approach Enabling Continuous User Requirements Engineering

Holger Fischer[1], Mirko Rose[1], and Enes Yigitbas[1]

[1]Paderborn University, s-lab – Software Quality Lab, Paderborn, Germany
{hfischer,mrose,eyigitbas}@s-lab.upb.de

Abstract. The digital transformation of traditional workflows challenges small and medium-sized and industrial enterprises likewise. Interactive systems have to be built to assist the people within their daily tasks as suitable as possible. Existing software engineering methods seems to be unsufficient because they don't consider end users in terms of active involvement during the development process and thus lack in the quality of the resulting product. In addition, digital transformation is an ongoing process and needs user participation as well as a continuous requirements refinement. Therefore, this paper describes an iterative task-driven approach to enable an incremental development based on a sustainable model that is continuously improved due to user reviews on runnable parts of the user interface.

Keywords: Task model, usability, human-centered design, requirements, continuity, sustainability, digital transformation

1 Introduction

Within the professional context human workflows get more and more software-based assistance. Digital offers and services are increasing and change also topics like communication, shopping, entertainment or health. This situation is called *'digital transformation'* and describes the economical challenges: Changing value chains and production processes as well as new business models and innovations. Strategies and change management are necessary to be part of the digital transformation. Nevertheless, project managers or IT departments cannot cope with the challenges on their own. Especially employees respectively end users have to be integrated in the change process due to the fact that they have to work with the assistance systems at last. The industrial internet of everything – known as 'Industry 4.0' – with its smart factories or computer-supported human workstations is one example where the employees will notice the impacts of the digital transformation. Workflows will not only be digitized, but changed in terms of new responsibilities or activities.

According to the World Quality Report [1], 1.560 CIOs and IT executives nominate *'Ensure end-user satisfaction'* along with *'Protect the corporate image'* and *'Increase quality awareness among all disciplines'* for the main three top quality objectives in the quality assurance of software products. *'Customer [or user] experi-*

ence' is also one of the five top influences on today's IT strategies. Thus, active user involvement has to be a major issue in a software engineering method within a digital transformation project.

However, transformation is a term that implies a fixed process with a defined beginning and end. Nowadays, it only takes short time periods to bring new software solutions onto the market and the progress is exponential. Therefore, the digitization has to be kind of a continuous evolution without a real end and will need a reusable requirements specification, e.g. formal models.

Software engineering (SE) as a discipline as well as SE methods (e.g. waterfall model [2], RUP [3], V-Model [4]) were built after the software crisis in the late 1960s. The objective is to ensure that software projects are developed with a defined range of functions, with high quality outcomes and within a specified budget and timeline. Despite all efforts, today's software products still lack of quality with regard to functionality and usability. According to the CHAOS manifesto [5], approximately 45% of the analyzed software projects have challenges within missing functionalities. In addition, business software in the European medium-sized enterprises lack on unused functionalities (36%) as well as on unusable software (21%) [6]. Thus, current SE methods seems to be unsufficient because they don't consider end users in terms of active involvement during the development process.

The challenge addressed in this paper is to establish a participatory task-driven SE method that integrates elicitation as well as evaluation steps with software users and takes their needs and expectations into account. Furthermore, the approach fosters iterations to identify implicit user needs and enables the development of software functionalities which are actually used instead of anticipated. Focusing on the sustainability within software development and continuity of requirements within the digital transformation of workflows, the approach uses the concept of model-driven development. The main focus of this paper concentrates on the requirements elicitation.

The paper is structured as following: First, the authors present previous approaches regarding SE methods with a focus on users, iterations or models and highlight potential for improvements. Then, the authors report on the concept of their approach. Finally, an initial evaluation of the concept and future work is described.

2 Related Work

Several different software development approaches exist, which either focus on user participation, iterative development or model-driven methods. Therefore, relevant approaches on human-centered design, activity-centered design, model-based user interface development and agility, which address parts of the described problem, are briefly summed up below.

Human-Centered Design (HCD) is an established methodology within software development with a focus on the users of a prospective system. HCD intends to create interactive solutions that match the users' needs and expectations as well as to support their tasks towards their specific goals. The advantages are far-reaching and include increased productivity, improved quality of work, and increased user satisfaction [7].

One of the central quality attributes for interactive systems is their usability [8]. The main standardization organizations (IEEE 98, ISO 91) have addressed this attribute for a long time [9]. Especially tasks that are either time-sensitive or security-critical benefit from user interfaces (UI), which are suitable, easily to understand as well as controllable. In order to create usable UIs, it is necessary to involve users from the early development stages of a project on. HCD focuses on different techniques applicable at different stages, e.g. contextual or behavioral inquiries (in terms of interviews, observations, etc.) to elicit the users' needs and specify the user requirements as well as prototyping and evaluations (e.g. paper mockups, usability tests) to identify further implicit requirements and to validate against the specified requirements.

Other approaches address the more specific paradigm of activity-centered design (ACD) and shift the focus from the users themselves to the roles they are working in, especially their responsibilities or activities. The user's behavior studied during interviews or observations is mapped to a role's activities and their comprised tasks [10]. Thereby, the analysts may identify breakdowns in the interaction with other roles as well as missing tasks. Thus, design solutions are built that support these tasks with functionalities, which are actually needed. Constantine & Lockwood explain in their approach of 'usage-centered design' [11] that "in the final analysis, understanding your users as people is far less important than understanding them as participants in activities". Hoekman [12] outlines four principles for the design of great software: Building only what is necessary, getting users up to speed quickly, preventing and handling errors, and designing for the activity. Nevertheless, there are still open challenges as the development will be completed after the interactive product is deployed. Due to the fact that the digital transformation is a continuous process, the concepts of the interactive product have to be stored more sustainable in ways of formalized representations, e.g. models.

Model-driven user interface development (MDUID) combines the two areas of model-driven software development (MDSD) and user interface development (UID) within one SE method. MDUID automatizes the development process of UIs by focusing on models instead of application code as a primary working base. Within MDUID, multiple UI models with different levels of abstractions are connected to stepwise transform the concept towards the final user interface (FUI) using model transformation. A unified reference framework for MDUID named CAMELEON reference framework (CRF) is described by Calvary et al. [13]. CRF divides the abstraction levels into 'task & concept', technology-independent 'abstract user interface' model (AUI), technology-specific 'concrete user interface' model (CUI) and the code-based FUI. Various modeling languages exists to cover the different abstraction levels. Both the 'model-based language for interactive applications' (MARIA XML) [14] and the 'interaction flow modeling language' (IFML) [15] provide a language and a tool for editing AUI models. A further approach for 'human-centered assessment and modeling to support task engineering for resilient systems' (HAMSTERS) is described by Martinie et al. [16]. The MDUID approaches described enable the specification as well as the support for generating initial code-based UIs. An open challenge exists within the lack of missing HCD activities [17] like iterating design solutions to elicit implicit user needs.

Agile model-driven development (AMDD) is an approach to specify models and implement solutions in an agile manner. Hence, it foster iterations of specifications as well as on solutions. Ambler [18] propose a method to work on models, "which are just barely good enough that drive your overall development efforts". This is done in contrast to MDSD approaches, where extensive models are created before starting to write source code. Therefore, the AMDD lifecycle begins with an envisioning stage to specify initial requirements and to sketch an initial architecture. After that, the actual iterations start using model storming, modeling iteration and test-driven development stages completed by an optional review of the developed increment. The iterations are repeated until a software product result meets the requirements.

In summary, all mentioned approaches have their reasons for existence and address specific topics in state-of-the-art software development. Nevertheless, there are still some open challenges. HCD includes lots of techniques to foster user participation, but it is still a discussion how to select appropriate techniques and how to integrate them on an operational level within existing software development processes. Thus, outcomes are mostly not directly convertible to software development due to their narrative textual representation. MDUID approaches start with models (e.g. task models) that specify a new system. Due to HCD the step of 'understanding and describing the context of use' before thinking about a new system is missing. In addition, existing concepts within MDUID, e.g. CTT or IFML, are technology-centered and hard to understand for end users. A preliminary step for describing the context of use together with end users is necessary. AMDD focuses on models to specify requirements, but these models aren't not transformable, e.g. to code. Thus, the approach is 'model-based' instead of 'model-driven'. Therefore, the authors' approach focuses on an iterative SE method that combines HCD, MDUID and AMDD to close these gaps.

3 Task-Driven Continuous Requirements Engineering

Taking the described challenges – user participation, iterations and model-driven development to foster continuous delivery due to digital transformation – into account, this paper presents the first step introducing a meta model of a flow model concept for requirements elicitation with user participation.

In order to define the concept of an integrated SE method, requirements have been specified to address the identified challenges. Therefore, available literature and especially remarks and field reports have been read. Thus, a model-driven approach have been sketched and stepwise elaborated to describe the dependencies between the different models used. Using the case study of an industrial project for an initial evaluation, the first stages of the concept have been assessed.

3.1 Requirements

Defining a continuous model-driven SE method requirements have to be specified to define the scope of the described approach. These requirements are categorized as follows.

- **User participation**
  - All stakeholders (people directly interacting with the system; people that are affected by the system; people that perceive itself to be affected by a decision, activity, or outcome) have to be involved in the software development lifecycle.
  - Stakeholders have to participate in every development stage starting with requirements elicitation.
- **Iterations**
  - User requirements have to be elicited over iterations to evoke implicit knowledge and user needs.
  - The amount of iterations have to depend on the quality of a functionality.
- **Task suitability**
  - Users' tasks have to be elicited properly as the system represents a tool to support the users in fulfillment of their tasks and goals.
- **Prototyping**
  - Software functionalities have to be validated using prototypes as early as possible.
- **Big Picture**
  - User requirements have to be synchronized with organizational business goals of the user's company and the other way around.
  - Dependencies of functionalities have to be identified in advance to allow an appropriate interaction and user experience.
- **Architecture**
  - Consistency over the user interface objects have to be ensured using reusable software components.
  - The systems' architecture has to be as flexible as possible to react on changed as well as on new added requirements.
- **Integration**
  - Human-Centeredness has to be an integrated aspect of a software engineering method rather than be a parallel activity.
- **Continuity**
  - Requirements have to be available in a formalized way to foster changeability and continuity with the ongoing digital transformation.

3.2 General Concept

HCD represents an approach for continuous user participation within the software development lifecycle to improve the system quality in terms of increasing the usability and user experience. The term 'design' describes the graphical layout and the specification of requirements and thus the priority of functionalities likewise, which are derived from explicit as well as implicit – often not verbalized within classical interviews – user needs. These needs result from a comprehensive insight and understanding of the users' tasks and goals as well as the social and physical working environment. Iterations of design solutions foster the stepwise elicitation and refinement of user requirements.

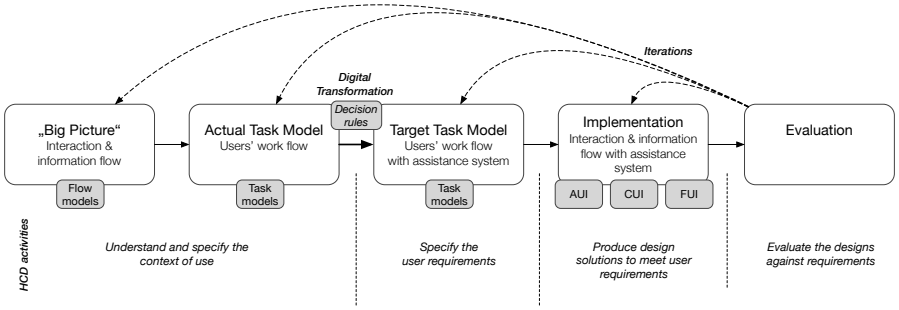

Fig. 1. Overall SE method

Hence, an iterative task-driven approach has been created (see Fig. 1). The approach starts with an envisioning stage to build an abstract "big picture" of participating roles, their responsibilities and interactions using the concept of flow models. These flow models are transformed into task models to divide activities down to operations. Using decision rules based on organizational (e.g. business goals, policies), legal (e.g. work safety) as well as human facets (e.g. fairness, controllability, information processing, technology self-efficacy), actual task models are transformed into the target task models that represents the digitized work flow with an assistance system. Then, these task models are transformed over AUI models into CUI models into the final code base (FUIs). Using the 'model-view-controller' (MVC) paradigm and a component based architecture (e.g. AngularJS[1], Polymer[2]), the application logic and the graphical design may be implemented in parallel. Design solutions could be evaluated and iterated if necessary.

3.3 Requirements Elicitation using Flow Models

Later changes that cause the modification of the system architecture need a lot of effort in terms of staff and costs. HCD will have an impact to reduce the later costs while expanding the analyzing stage in the run-up to the project. In addition, it will have an impact on the sustainability of requirements within a digital transformation. To make the interdependencies between people, work items and existing systems visible, this paper propose the concept of flow models (adapted from [19]; see Fig. 2) for the initial envisioning with user participation. Thus, an overview or "big picture" is created on an abstract level to identify dependencies between possible prospective functionalities or workflows and to support the specification of the system architecture likewise. Flow models are used to describe the actual situation – instead of specifying the target situation – and could be build during interviews, workshops or observations with customers and users. The specification of the prospective situation will be part in the next step within the task model concept. The differences between flow

[1] AngularJS: https://angularjs.org (Last view: January 2016)
[2] Google Polymer: https://www.polymer-project.org/1.0/ (Last view: January 2016)

and task models exist within their abstraction level and complexity. Flow models are a first step to be worked out with users. They are barely formal enough to be transformable to the more complex task models using model transformation.

The major concept of a flow model is the '*role*'. Each person interacting with a system within a working or business context is doing this within its role. Each role has '*responsibilities*' – activities it is responsible for – and belongs to a '*stakeholder type*', which represents its closeness to the system. It is directly affected by the system or it is implicit affected by the outcome of the system. Roles are connected with a '*communication flow*' either to another role or a '*place*'. A place could be a physical one (e.g. notice board) or a virtual one (e.g. information system). Each communication flow has a topic and transfers an '*information*', which consists either of a single value or of a more complex '*work item*', e.g. a collection of information like a document. Furthermore, '*annotations*' highlight important aspects that are crucial for a communication while '*breakdowns*' highlight kinds of problems, challenges or disruptions (e.g. changing media) within interactions or within work items and thus represents possibilities for digitization and assistance.

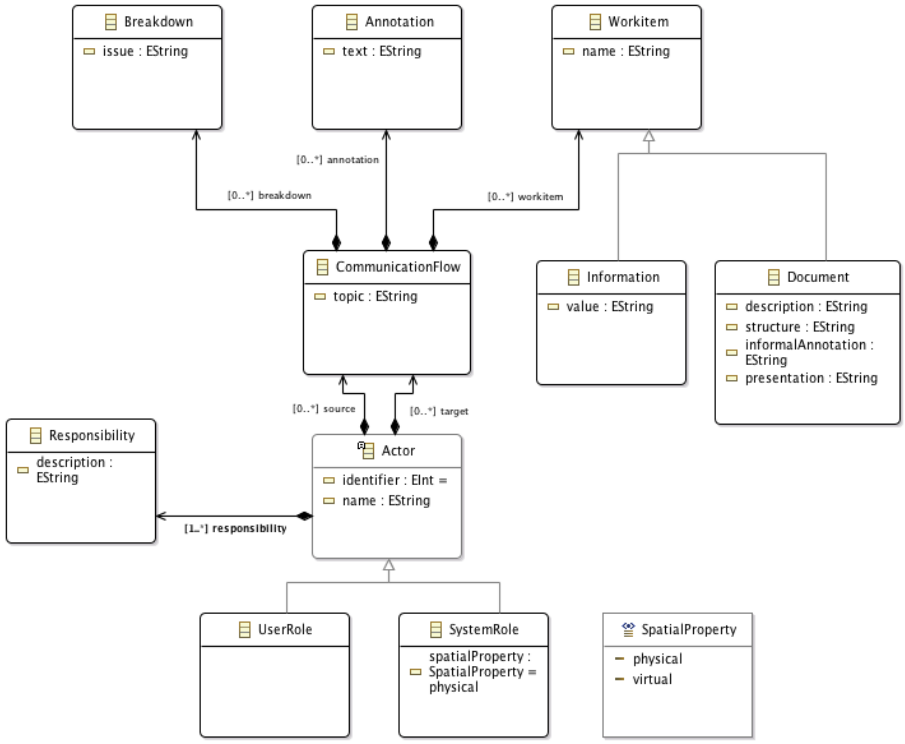

Fig. 2. Meta model of flow model

3.4 Continuous Task Model Refinement

The flow models described above represent the base for further refinements. Therefore, especially the roles' responsibilities will be transformed into tasks of task models to divide human activities. Furthermore, communication flows represent triggers to break tasks down to temporal canonical operations. These operations represent human actions (enter, change, trigger, select, inform) that are mostly based on cognitive analysis and decision. Thus, it is possible to get a task suitable user interface while matching these operations towards interactions objects.

To specify appropriate task models, the HAMSTERS notation [16] is used due to the context of resilient systems. An alternative notation exists within the simpler concur task tree (CTT) concept [20].

Having the model-driven approach in mind, task models represent a possibility to foster a continuous requirement engineering. Thinking about the (semi) automation of model-to-model and model-to-code transformation, it will lead to a model base where evaluation results as well as new requirements change the task models and generate modified UIs. This step of the concept is currently in development, but the envisioning stage has been already investigated within a current practical project.

4 Case Study

A current cooperative project with a medium-sized enterprise (SME) was used to evaluate the suitability of flow model as a first part of the overall approach.

The SME is active within the context of building construction and supports their customers (e.g. architects, engineers, constructors) with free of charge services in order to convince them to buy their building elements later on. Some stand-alone solutions exist that supports the involved people with some assistance. With the aim to improve the overall construction planning and coordination process and to connect existing solutions, a project has been set up to analyze the context of use within the company (sales, engineering, consulting) as well as with the users of their services.

The analysis in the project includes about 12 interviews. Each interview has two interviewee working in the same department and two to three interviewers. The activities of the interviewers are as follows: Leading the interview, taking notes and directly creating a flow model within the interview. In some cases, the person asking the questions and creating the model was the same. The flow models were created using paper cards with different forms and colors. Afterwards, the different flow models were validated with the notes taken, digitized on a computer, consolidated into one bigger flow model and used for a workshop where all interviewee were participating to discuss.

As a first result, the paper based flow models were suitable for the creation within the interview. The information became visible for everyone and even the interviewee started to pick up a card and telling their stories with it, because the 'prototyping character' motivated them to do so. They were proud to talk about their work.

Creating flow models within interviews or in workshops with multiple participants have advantages as well as disadvantages and depend on the project structure and

timeline. You will get one bigger flow model based of a group consensus during a workshop, but “louder” participants with more self-confidence could enforce their view within the model. Having multiple interviews with single people will amount to multiple smaller flow models. These will reflect more personal views of each participant. Thus, you will have to consolidate the flow models afterwards and to discuss particularly validate the result within an additional workshop.

Furthermore, sketched up breakdowns and annotations served as clues for prospective assistance through a digital system. They were used in the second part of the interviews to talk about future visions.

5 Conclusion

In this paper, the lack of quality within today’s interactive systems due to unsufficient software engineering methods, which don’t consider end users in terms of active involvement during the development process, and the rigidity of requirements in the context of digital transformation have been discussed. Therefore, an approach has been presented that uses human-centered design to focus on the users’ workflow and take model-driven software development into account to create a flexible sustainable base for continuous requirements engineering. The concept of flow models has been introduced to sketch up the initial abstract model within the envisioning stage.

It has been shown that visual representations of models lead to an open communication of all involved roles as well as on the differences in their awareness. Thus, actual users of a prospective system could already be integrated from the very beginning in a software project and it will foster the digital transformation of workflows due to participation.

In our future work we will further expand the concepts in more detail. In a next step the model-to-model transformation from flow models to task models as well as the ‘digital transformation’ between actual and target task models will be specified. After that, it will be investigated how the concept of the CAMELEON reference framework fits for the transformation from task models to AUI models to CUI models to the code base of the FUI.

Furthermore, it will be analyzed how general UI guidelines or company corporate design guidelines could already be integrated in the transformation of FUIs – partly shown by Yigitbas et al. [21].

All steps of the approach will be evaluated within further software projects as well as within teaching activities.

References

1. Buenen, M., Walgude, A.: World Quality Report 2015-16, 7th edition. Capgemini, Sogeti and HP (eds.). https://www.capgemini.com/thought-leadership/world-quality-report-2015-16 (2015)
2. Royce, W.: Managing the Development of Large Software Systems. In: Proceedings of IEEE WESCON, pp. 1-9 (1970)

3. Kruchten, P.: The Rational Unified Process – An Introduction. Addison-Wesley, Amsterdam (1998)
4. V-Modell XT. http://v-modell.iabg.de/v-modell-xt-html-english/index.html (Last view: January 2016)
5. Johnson, J.: CHAOS 2014. The Standish Group (2014)
6. Sage Software GmbH. Independent Study on IT investments. http://www.sage.de/header/presse/pressemitteilungen/2014/05/19/sage-studie-europaeische-firmen-verschwenden-milliarden-euro-durch-ungenutzte-software (2014) (Last view: January 2016)
7. Jokela, T.: An Assessment Approach for User-Centred Design Processes. In: Proceedings of EuroSPI 2001. Limerick Institute of Technology Press, Limerick (2001)
8. Bevan, N.: Quality in Use – Meeting User Needs for Quality. In: Journal of System and Software, pp. 89-96. Elsevier Science Inc., New York (1999)
9. Granollers, T., Lorès, J., Perdrix, F.: Usability Engineering Process Model. Integration with Software Engineering. In: Proceedings of the Tenth International Conference on Human-Computer Interaction, pp. 965-969. Lawrence Erlbaum Associates, New Jersey (2002)
10. Norman, D.: Human-Centered Design Considered Harmful. In: interactions – Ambient intelligence: exploring our living environment, Vol. 12 (4), pp. 14-19 (2005)
11. Constantine, L.L., Lockwood, L.A.D.: Software for Use: A Practical Guide to the Models and Methods of Usage-Centered Design. Addison-Wesley, Boston (2011)
12. Hoekman, R.: Designing the Obvious: A Common Sense Approach to Web and Mobile Application Design, 2nd edition. New Riders, Berkeley (2011)
13. Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J.: A Unifying Reference Framework for Multi-target User Interfaces. In: Interacting with Computers, vol. 15, pp. 289-308. Elsevier, New York (2003)
14. Paternò, F., Santoro, C., 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)
15. Brambilla, M., Fraternali, P.: Interaction Flow Modeling Language – Model-Driven UI Engineering of Web and Mobile Apps with IFML. Elsevier Inc., Waltham (2015)
16. Martinie, C., Navarre, D., Palanque, P.: A Multi-Formalism Approach for Model-Based Dynamic Distribution of User Interfaces of Critical Interactive Systems. In: International Journal of Human-Computer Studies, vol. 72 (1), pp. 77-99. Academic Press, Duluth (2014)
17. ISO 9241-210 Ergonomics of Human-System Interaction – Human-Centered Design for Interactive Systems (2010)
18. Abler, S.W.: Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development. http://agilemodeling.com/essays/amdd.htm (Last view: January 2016)
19. Beyer, H., Holtzblatt, K.: Contextual Design – Defining Customer-Centered Systems. Morgan Kaufmann, San Francisco (1997)
20. Paternò, F., Santoro, C., Spano, L.D., Raggett, D.: MBUI – Task Models. W3C Working Group Note 08 April 2014. http://www.w3.org/TR/task-models/ (2014) (Last view: January 2016)
21. Yigitbas, E., Mohrmann, B., Sauer, S.: Model-driven UI Development Integrating HCI Patterns. In: Proceedings of the International Workshop on Large-scale and model-based Interactive Systems: Approaches and Challenges at EICS'15. CEUR Workshop Proceedings, vol. 1380 (2015)