Advances in Engineering Software 40 (2009) 1223–1230

Contents lists available at ScienceDirect

Advances in Engineering Software

journal homepage: www.elsevier.com/locate/advengsoft

Integration of usability evaluation and model-based software development

Stefan Propp *, Gregor Buchholz, Peter Forbrig

University of Rostock, Institute of Computer Science, Albert Einstein Str. 21, 18059 Rostock, Germany

ARTICLE INFO

Article history:
Received 22 September 2008
Received in revised form 19 November 2008
Accepted 19 January 2009
Available online 9 April 2009

Keywords:
Model-based usability evaluation
Task models
HCI

ABSTRACT

Model-based software development is carried out as a well defined process. Depending on the applied approach, different phases can be distinguished, e.g. requirements specification, design, prototyping, implementation and usability evaluation. During this iterative process manifold artifacts are developed and modified, including, e.g. models, source code and usability evaluation data. CASE tools support the development stages well, but lack a seamless integration of usability evaluation methods. We aim at bridging the gap between development and usability evaluation, through enabling the cooperative use of artifacts with the particular tools. As a result of integration usability experts save time to prepare an evaluation and evaluation results can be easier incorporated back into the development process. We show exemplary our work on enhancing the Eclipse framework to support usability evaluation for task model-based software development.

1. Introduction

Model-based software development describes a process, which starts with abstract models and refines them iteratively. Models are transformed and adapted to more concrete models for different platforms and finally transformed into code which is deployed at the target platform. Traditional usability evaluation provides methods to evaluate a couple of different artifacts, but lacks a seamless integration into the development process from early development stages to the deployment of the product. Therefore we aim at bridging the gap between development and usability evaluation, through enabling the cooperative use of artifacts with developers and usability experts and pave the way towards a seamless integration of both processes.

Our work specifically focuses on the task model-based development of interactive systems [2]. Task models describe possible sequences of sub tasks to accomplish a certain goal and temporal relations between tasks [7]. Task models serve as basis for the model-based development of user interfaces. To evaluate the usability of the derived user interface the underlying task model can be used to track the user interaction at a task-based level of abstraction. The particular advantage is a concise overview, which hides the device specific user interface events. Therefore the analysis of a task trace is better readable and traces through an application deployed at different devices can be compared directly. Examples using this technique are RemUSINE [8] and ReModEl [1]. RemUSINE firstly captures the user interactions and subsequently maps the interactions to a trace of tasks, which were carried out. However ReModEl deploys a task engine which natively recognizes the tasks at runtime and captures the tasks without mapping stage. For the subsequent analysis of task traces, Malý and Slavík suggested a visualization technique [5], further developed in [8,9]. A task trace is depicted as timeline to compare different users' behavior at a task-based level.

The previously stated approaches serve as individual methods and target deployed applications, but are not integrated into the development environment. This paper discusses our approach of integration to provide usability testing at all stages of the development instantly on the artifacts currently under development. A prototypical implementation of the described usability evaluation tool support as extension to an existing task interpretation engine developed at our working group [2] is presented as well. This implementation covers the creation, simulation modification and refinement of task models and provides support in developing dialog graphs and graphical user interfaces based upon pattern-driven UI development [11]. Details of the interpretation engine are given in Section 2.2.2. Evaluations as described here are intended to seamlessly fit into the development process. Within these evaluations, the focus lies on the efficiency of the system developed so far. The approach is discussed in Section 2. The integration into the development tools implemented as plugins for the Eclipse IDE is exemplified in Section 3.

2. Concept

2.1. Model-based usability evaluation

From our perspective it is a main principle of model driven software development (MDSD), that both software engineers and user

* Corresponding author. Tel.: +49 381 498 7444; fax: +49 381 498 7482.
E-mail address: stefan.propp@uni-rostock.de (S. Propp).

0965-9978/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2009.01.025

interface designers base their work on the same models. Progress in development is reflected in a sequence of transformations from the initial models (task model, user model, device model) as results of the requirements analysis to a finished software product. Those transformations cannot be done in a fully automated way but require humans using interactive tools. In the past we have developed such tools providing support in the creation and transformation of models [2,11]. Initially, a hierarchical task model describes the decomposition of the root task (also referred to as the user's goal) into sub tasks that are to be carried out to reach the goal. Tasks can be optional and iterative and temporal relationships between different sub tasks define the order in which the sub tasks have to be executed. The user model describes the different roles a user can take on and by assigning roles to sub tasks cooperative behavior can be modeled where different users are involved in the work on a task (e.g. a customer and an employee of a travel agency take part in the task "book flight"). Device models specify the capabilities of different types of devices the software product is planned to be used at. They are also structured hierarchically. Hence an include-relationship can be modeled (for example, if the sub task "seat reservation" is specified to be executable on a mobile phone it can be performed on a PDA as well). Based on those models the development of a user interface comprehends the creation of a device independent abstract user interface (AUI), one or more device specific concrete user interfaces (CUI) specified by dialog graphs [2] and the final interfaces as device specific rendering results. The integration of usability evaluation into this process of transformations is depicted in Fig. 1.

Each development stage is accompanied by usability evaluation tests (depicted as Simulation 1–4 in Fig. 1):

1. Task model simulations are used to reveal structural problems in the underlying model. These problems may be wrong or not appropriate temporal relations between tasks and inadequate arrangements and groupings in the task hierarchy. Interactive simulations can further uncover lacking understanding of the task domain and indicate modeling mistakes. Task models in this stage do not necessarily contain tasks that have to be carried out as interactive tasks only but may also cover activities that are executed without any support by the system under development. The purpose of evaluations in this stage is to fortify a common consensus about what processes the system is to support and how these processes are structured.
2. Evaluation techniques targeting the dialog model allow finding deficient support of the user in providing access to the modeled tasks at the right time and the right place. While the dialog model used in this stage does not include any UI elements the evaluation is conducted with a very basic representation of the tasks arranged in the dialogs. Thus, the navigational structure of the system is verified. The simulation may be carried out by an expert based on scenarios defined earlier in the development process or by potential users who are instructed to simulate the execution of tasks necessary to accomplish a certain goal.
3. Since abstract UIs are automatically generated from the dialog model, they represent first prototypical and simple UIs. Testing facilitates the pattern-based process of replacing very basic UI components with more specific ones, comparing the efficiency that is reached by executing tasks while using either one or another version of the interface. The iterative sequence of refining the GUI and evaluating it starts with the button representation of tasks used as described before. In contrast to evaluations in phase 2 now the refined user interface components are focused regarding their contribution to a usable way of fulfilling a specific task. Up to now, there is hardly any functionality of the system implemented. Interface elements may be linked to a prototypical or static data model but there is no underlying logic.
4. Once the evaluated interface is merged with the application model and a system offering the entire proposed functionality, further evaluation methods based on UI events, video recordings and other techniques can be applied to ensure and optimize the final UI's usability. All these additional sources of evaluation data have to be linked to corresponding elements in the task model. Thus, after finding usability problems in this stage the techniques described in phase 1–3 can be applied focusing the basic cause of the problem just found, if necessary.

The next sections will go into the details of model-related evaluation techniques in the successive development stages as outlined before.

2.2. Capturing interactions

2.2.1. Capturing at different levels of abstraction

A common used technique of usability evaluation is to equip the test users with devices and applications that have to be tested and observe the interaction with the device [6]. The observed data comprises for instance video and audio sequences, mouse movements, keystrokes or annotations by an expert. In ubiquitous environments, where users are moving and interacting with different devices' UIs, some additional sensors are beneficial, for instance location sensors and information about the handled objects in the environment (e.g. exploiting RFID information). We focus on interaction events, which can be captured automatically by the system. Such a trace of interactions can be captured at different levels of abstraction [4]: beginning with physical events (e.g. hand moving a mouse), over UI events (e.g. button press), to goal-related information (e.g. printing a document). The lower levels of abstraction can be captured simply as system internal events, but cause a vast amount of data, whereas higher level events can be interpreted more directly, but normally need to be derived from the lower level events. For deriving higher level events, we aim at reusing the models from software engineering (depicted in Fig. 1), which are task, user, device and domain model.

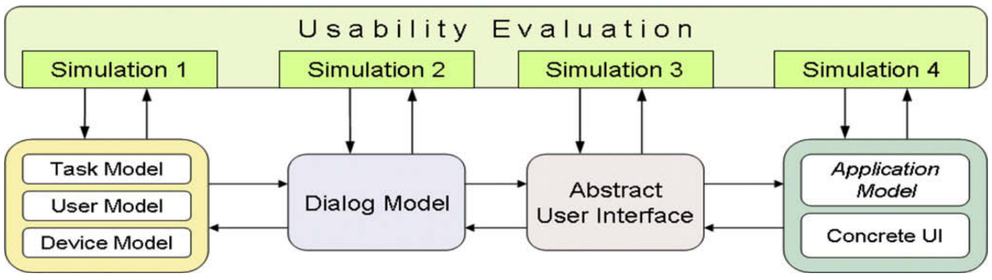

Fig. 1. Integration of usability evaluation into MDSD.