Bridging the Gap between Model and Design of User Interfaces

Sebastian Feuerstack, Marco Blumendorf, Sahin Albayrak

DAI-Labor
Technische Universität Berlin
Secretary GOR 1-1, Franklinstrasse 28/29
D-10587 Berlin, Germany
Sebastian.Feuerstack@dai-labor.de
Marco.Blumendorf@dai-labor.de
Sahin.Albayrak@dai-labor.de

Abstract: The creation of user interfaces usually involves various people in different roles and several tools that are designed to support each specific role. In this paper we propose a tool for rapid prototyping that allows all parties involved to directly interact with the system under development. The tool is based on task tree development and integrates the system designer, the user interface designer, the usability expert, and the user interface developer in a common process. The final system is derived from two sources, the task model specified by the system architect and the final user interface specified by the user interface developer and designer. Aggregating the runtime system and the design tools into one complete integrated system is our approach to bridge the gap between the user interface designer working on system mock-ups and the actual developers implementing the system.

1 Introduction

The development of high quality user interfaces is not an easy task, usually involving several people with different skills, knowledge and professional backgrounds. Various prototypes on different levels of the system (backend, UI design, UI implementation) are designed and considered for the development of the system, sometimes with support by different tools. In this paper we propose an approach that consequently utilizes the runtime view of the system being developed to provide an integrated development and testing environment supporting all roles involved in the development process. Our approach helps bridging the gap between the work of the designer, the usability expert, and the developer by providing a common web-based system that allows all parties to jointly work with the system under development during the creation of the user interface.

In the following chapter we provide a brief overview of the current state of the art, followed by a description of our approach in chapter 3. We then describe the anticipated development process (chapter 4) and the system we created to realize our approach (chapter 5). We then conclude and summarize our work and discuss some future work.

2 Related Work

The specification of a system using task trees as a basic format is a well accepted approach for interactive systems [Ca03]. It helps identifying the main tasks of an interactive application starting at a very abstract design level and moving to a more concrete one step by step. The elements in the task model therefore represent actions the user can undertake, actions the system undertakes and temporal relationships between these actions. In [Pa99], Fabio Paternò proposed the Concurrent Task Tree (CTT) notation to formalize the specification of the system.

The development of high quality multi-modal user interfaces is still a big and time-consuming challenge, as it requires experts of different topics to jointly work together. It often results in extensive iterations of development cycles and because of dependencies between the results of the involved parties it usually takes a long time until the first prototype is ready for usability tests. Several systems have already been proposed utilizing the task tree definition as starting point for the further development of user interfaces. Two different approaches can be distinguished here: The first one uses the task tree as a basis for user interface development tools, supporting the specification of the user interface based on the systems task tree as done in IdealXML [Mo06], DiaTask [Re04], Theresa [Mo04] or OlivaNova [Es06]. The second approach uses the task tree as the basic system description at runtime which defines the runtime behaviour of the system by identifying the Enabled Task Set (ETS) [Pa99] (active tasks) and the transitions between the Enabled Task Sets [Pa06][Co03][Kl05]. In contrast to very interesting approaches that focus on rapid prototyping like the deriving of widgets from paper sketches [Co05] to speed up the design to code step, the system presented in the following focuses on a user interface development process supporting all involved parties.

3 Approach

In our approach we use the CTT notation by both the user interface development process and the runtime environment to realize a collaborative user interface design process. Therefore we embed a user interface development process into a runtime environment for the provisioning of multi-modal user interfaces. A dynamic web-page presents the graphical user interface of the application and also serves as development environment.

Concur Task Trees that have been identified in the analysis phase of the design process provide the basic user interface structure and are evolved step by step within the runtime environment to the final system in a collaborative effort. This approach eliminates the need for extensive development iterations because of dependencies between the involved parties and enables rapid prototyping making intermediary results instantly available for usability testing.

In this paper we present a scenario for a cooking assistant interactively guiding a user through the cooking process as running example to describe our approach. The cooking assistant can be accessed through a graphical user interface (web browser) or a voice-based interface (voice xml) or in a multimodal way through a combination of both.

4 Developing User Interfaces at Runtime

The creation of a user interface usually takes place in five iterative phases, which we have embedded into an integrated collaborative process: analysis, design, implementation, usability testing and deployment.

In the first step, the analysis of the anticipated system has to take place in form of a task analysis. The result of this phase is a task tree of the system specified in the CTT notation. This task tree forms the basis of the application to be developed, defining the tasks the user will carry out through the user interface (interaction tasks), the tasks that have to be accomplished by the backend system (system tasks) as well as the temporal relationships between the tasks. Figure 1 presents a task analysis of the virtual cook application using the CTTE editor [Mo02].

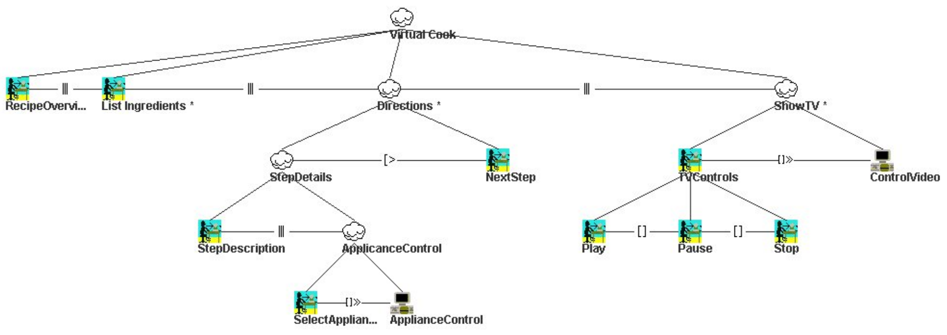


Figure 1: Concur Task Tree of the virtual cook application. Cloud symbols specify abstract tasks, the computer symbol a system task and the working person an interaction task.

As soon as the tree is specified, the system can already be simulated and the implementation can be started at two points: backend and frontend. The system developer can connect the system tasks to the backend logic and the user interface designer can propose a layout for each view by uploading an image or voice sample for each task. Thus, the designer already considers the structure and the task descriptions for his image (or voice) based design and can immediately check how the design supports the whole system.

A usability expert can continuously monitor the evolving prototype and suggest improvements to the designer. It is also possible to involve users into the evaluation at this early stage, by running guided tests where the users can explore specific aspects of the system. As soon as the usability expert marks a presentation of an enabled task set as final, a user interface developer can replace the composed images of the task sets with native code of the final user interface. The developed user interface implementation can then be reviewed by the designer and the usability expert to ensure that it matches the original design and the usability requirements.

The described process is suitable for the evolutionary creation of multi-modal user interfaces. We distinguish two multimodal approaches: *simultaneous multimodality*, supporting the usage of the application through multiple coherent modalities and *sequential multimodality*, providing access through different independent modalities. The creation of sequential multimodal user interfaces is supported by the same process as the creation of graphical user interfaces. Instead of using images it is possible to link tasks to recorded audio files for prototyping. However, prototyping of simultaneous multimodality is much harder, but can be solved by providing animations that synchronize video with a recorded voice output as a prototype.

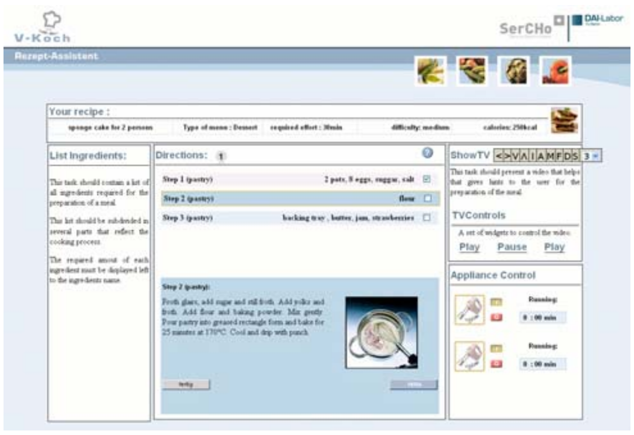
To support the process as described here we implemented a runtime system based on the definition of the task tree, providing the infrastructure necessary to allow the collaborative development of the user interface in an integrated process, combining the needs of the different roles involved in the development.

5 A CTT-based Runtime System for Multimodal User Interfaces

In order to realize the process as described in the previous section, we built a runtime system, allowing all involved parties to work simultaneously on the different aspects of the user interface. To allow continuous access the system runs as a web application, providing browser-based access at all development states. Based on the task tree defining the basic structure of the application, loaded into the runtime system as initial step, the first set of active tasks (Enabled Task Set) as defined by the temporal relationships between the tasks is automatically derived. From each Enabled Task Set (ETS) that defines a possible state of the user interface we can automatically derive an Abstract User Interface (AUI) as defined in [Ca03]. The AUI defines the structure of the final user interface and serves as basis for the layout of the application's user interface. The AUI basically presents each task of an active task set as a frame, providing a basic version of the system allowing simulating the interaction steps. Similar to [Mo06] all enabled tasks are presented simultaneously to the user and the knowledge about the tree hierarchy of the ETS defines the nested structure of the user interface. Each task in this basic structure (not only the leaf nodes as in other approaches [Pa06]) can now be annotated with additional presentation information allowing deriving the final user interface.

In comparison to approaches incorporating other specialized models like [Ca03] only one model has to be managed at runtime and no mapping between models is necessary. A downside is the reduced flexibility when attempting to adapt user interfaces to different platforms. The fact that not only the leaf tasks but also higher level tasks have representations in our approach, allows us to incorporate the original task hierarchy into the final user interface using include mechanisms.

Our runtime system loads the complete task tree including all annotations into our task tree interpreter. The interpreter then identifies the initial ETS and utilizes our include mechanism to assemble the final user interface (see figure 2). In difference to already existing approaches like [Mo06][Pa02] we do not distinguish between a tool and the final runtime environment of the application. Each modification directly effects the resulting application. This WYSIWYG approach allows all involved parties (user interface designers, user interface developers, usability experts and service developers) to work simultaneously as soon as a first sketch of a task tree has been realized. Each developer can work independently according to the process we have described, but the centralized web-based development tool allows the coordination of the different work packages and promotes the exchange of feedback, based on a commonly understood and available prototype. Every person involved in the user development process can instantly check the effects of any modification on the whole application. This system features an integrated development process as well as the delivery of user interfaces based on a task model.



Your recipe :

sponge cake for 2 persons | Type of menu : Dessert | required effort : 30min | difficulty: medium | calories: 250kcal

List Ingredients:

This task should contain a list of all ingredients required for the preparation of a meal.

This list should be subdivided in several parts that reflect the cooking process.

The required amount of each ingredient must be displayed left to the ingredients name.

Directions: 1

Step 1 (pastry) 2 pots, 8 eggs, suggar, salt

Step 2 (pastry) flour

Step 3 (pastry) backing tray, butter, jam, strawberries

Step 2 (pastry):

Froth glairs, add sugar and still froth. Add yolks and froth. Add flour and baking powder. Mix gently. Pour pastry into greased rectangle form and bake for 25 minutes at 170°C. Cool and drip with punch.

ShowTV < > V Λ I A M F D S 3

This task should present a video that helps that gives hints to the user for the preparation of the meal.

TVControls

A set of widgets to control the video.

Play Pause Play

Appliance Control

Running: 0 : 00 min

Running: 0 : 00 min

Figure 2: Example of the virtual cook user interface loaded into the runtime environment.

Figure 2 shows an example of the virtual cook user-interface during the development process. The task tree has been loaded into the runtime environment and each task that is part of the enabled task set is displayed as a box on the screen. The runtime environment considers the parent tasks of the task tree as well and renders these relationships as nested boxes.

Some boxes of the virtual cook user-interface shown in figure 2 are already implemented (e.g. the box “Directions” in HTML), designed as an image (boxes “Your recipe” and “Appliance Control”) or just contain a textual description of the task (“List ingredients” and “TV-Monitor”). Each box includes the name of the associated task, the task description and a set of buttons that are used to rearrange the boxes that are on the same abstraction level (“<”,”>”,”^”,”v”,) to upload an image (I), audio (A) or video (M) file, to mark a design as final (F), or to edit the task description (D) or the source code (S) that reflects the image for the final user interface. Additionally a navigation path through the boxes can be set by using a popup that allows setting a number used for ordering the tasks. The ordering is most important for a voice presentation of the user interface but can be used to define a tab sequence in a graphical user interface as well.

Our runtime environment is realized using an interpreter for the CTTE xml format of concurrent task trees. The interpreter controls JSP pages in a web application container and allows a direct manipulation as we described above. We have chosen JSP-based generation of user interfaces because it is a broadly accepted technology and there is a lot of tools available supporting JSP page development. Another important aspect was that the JSP format already has features for including sub-pages, which made it easy for us to realize the aggregation of boxes reflecting the CTT tree hierarchy.

6 Conclusion and Outlook

Our integrated solution for the development and design of user interfaces bridges the gap between the design of user interfaces and the implementation of the actual system. We directly involve the designer and the usability expert in the development process and support the creation of a first draft of the final user interface based on task trees annotated by design specifications of the user interface designer.

An advantage of our approach is the possibility to create multimodal user interfaces based on the described ideas. Instead of just creating graphical designs for the user interface, the different tasks can also be annotated with additional voice output or animations, which can be provided to allow multimodal enhancements of user interfaces.

We have experienced that finding the optimal abstraction level during the task-based analysis is still an open problem, as it directly affects the granularity and thereby the interactivity of the resulting image based prototype. We are currently planning to enhance our approach by adding access to a design pattern library to speed up the image to code step that has to be done by the user interface developer.

In the future we want to exploit the possibility to read the CC/PP profile of devices accessing the system already built into our runtime system to emphasize support for multi-platform UI’s as described in [Ri05]. A problem here is the increased workload to create the user interface, which we attempt to compensate by deriving the UI model from the design process of the user interface. The designer specifies one or more UI’s and the required work is reduced more and more, as the model defining the commons between the UI’s and the meaning of the UI parts evolves.

The system presented offers an approach supporting an evolutionary process to develop user interfaces providing an environment allowing reviewing and discussion of results in a coordinated process. First evaluations of the approach showed that the common system view shared by all involved parties simplifies communication and allows the easier integration of results which results in increased productivity when developing user interfaces in a team.

7 References

[Ca03] G. Calvary et all: A unifying reference framework for multi-target user interfaces. In *Interacting with Computers*, Vol. 15, No. 3. (June 2003), pp. 289-308.

[Co03] K. Coninx, K. Luyten. C. Vandervelpen, J. Van den Bergh, B. Creemers: Dygimes: Dynamically Generating Interfaces for Mobile Computing Devices and Embedded Systems. In *Human-Computer Interaction with Mobile Devices and Services, 5th International Symposium, Mobile HCI*, p. 256-270, 2003

[Co05] A. Coyette, J. Vanderdonckt, A Sketching Tool for Designing Anyuser, Anyplatform, Anywhere User Interfaces. In *Proceedings of 10th IFIP TC 13 Int. Conf. on Human-Computer Interaction Interact'2005*, Lecture Notes in Computer Science, Vol. 3585, Springer-Verlag, 2005, pp. 550-564.

[Es06] S. Espana, I. Pederiva, J. I. Panach, Ó.Pastor: Integrating Model-Based and Task-Based Approaches to User Interface Generation. Accepted for *CADUI 2006*.

[Kl05] T. Klug, J. Kangasharju: Executable task models. In *Proceedings of the 4th international Workshop on Task Models and Diagrams*, pp. 119-122. TAMODIA '05, ACM Press.

[Mo02] G. Mori, F. Paternò, C. Santoro: CTTE: support for developing and analyzing task models for interactive system design. *IEEE Trans. Softw. Eng.* 28, 8 (Aug. 2002), pp. 797-813.

[Mo04] G. Mori, F. Paternò, C. Santoro: Design and Development of Multi-Device User Interfaces through Multiple Logical Descriptions. In *IEEE Transactions on Software Engineering*, pp. 507-520, 2004.

[Mo06] F. Montero and V. López-Jaquero: IDEALXML: An Interaction Design Tool and a Task-based Approach to User Interface Design. Accepted for *CADUI 200*6.

[Pa99] F. Paterno: Model-based Design and Evaluation of Interactive Applications. Springer Verlag. Berlin 1999.

[Pa02] F. Paternò, C. Santoro: One Model, Many Interfaces. In *Proceedings of CADUI 2002*, pp. 143-154.

[Pa06] F. Paternò, I. Santos: Designing and Developing Multi-User, Multi-Device Web Interfaces. Accepted for *CADUI 2006*.

[Re04] D. Reichart, P. Forbrig, A. Dittmar: Task models as basis for requirements engineering and software execution. In *Proceedings of the 3rd international Workshop on Task Models and Diagrams*, pp. 51-58. TAMODIA '04, ACM Press

[Ri05] A. Rieger, R. Cissée, S. Feuerstack, J. Wohltorf, S. Albayrak: An Agent-Based Architecture for Ubiquitous Multimodal User Interfaces. *International Conference on Active Media Technology*, Takamatsu, Kagawa, Japan, 2005