Multimodal User Interfaces for Smart Environments: The Multi-Access Service Platform

Marco Blumendorf, Sebastian Feuerstack, Sahin Albayrak
DAI-Labor, TU-Berlin
Ernst-Reuter-Platz 7, D-10587 Berlin
{Marco.Blumendorf, Sebastian.Feuerstack, Sahin.Albayrak}@DAI-Labor.de

ABSTRACT
User interface models are a well accepted approach to handle increasing user interface complexity. The approach presented in this paper utilizes user interface models at runtime to provide a basis for user interface distribution and synchronization. Task and domain model synchronize workflow and dynamic content across devices and modalities. A cooking assistant serves as example application to demonstrate multimodality and distribution. Additionally a debugger allows the inspection of the underlying user interface models at runtime.

Categories and Subject Descriptors
H.5 [**Information Interfaces and Presentation**]: User interfaces; D.2.2 [**Software Engineering**]: Design Tools and Techniques-*User Interfaces*; H.1.2 [**Models and Principles**]: User/Machine Systems-*Human factors;* H.5.2 [**Information Interfaces and Presentation**]: User Interfaces-g*raphical user interfaces, interaction styles, input devices and strategies, voice I/O.*

General Terms
Design, Human Factors

Keywords
Model-based user interfaces, runtime interpretation, Smart home environments, ubiquitous computing, multimodal interaction, human-computer interaction, interface design, usability

1. INTRODUCTION
Ambient environments comprising numerous networked interaction devices challenge interface developers to provide approaches that exploit these new capabilities. In this paper we describe an approach that addresses the need to adapt the interface to the environment. A runtime system, utilizing user interface models supports multimodal interaction and user interface distribution. The next section gives an overview of the developed system, followed by the description of an example, demonstrating the features.

2. THE MULTI-ACCESS SERVICE PLATFORM
The Multi-Access Service Platform (MASP) is a runtime system we created to address deployment and runtime issues when developing interaction in smart environments. The system focuses on multimodal applications and follows a model-based approach. Based on a user interface model, the system allows controlling multiple user interfaces and is able to deliver the partial UI artifacts to different devices supporting different interaction modalities. Based on the runtime interpretation of the model the MASP is able to synchronize the distributed parts of such user interfaces (UIs).

The underlying user interface model is based on the ideas of the cameleon reference framework [2] and similarly separates multiple levels of abstraction. A task- and domain model define the workflow and dynamic data of the application, providing the basic information required for the interaction. The actual user interface is defined via templates providing final UI code (i.e. HTML and VoiceXML).

The task tree [4] defines the application workflow using the Concurrent Task Tree (CTT) notation [5]. Similar to [3] this allows assembling user interfaces from multiple parts based on the enabled task set. Objects, related to the identified tasks are defined as domain model allowing the exchange of information between tasks and with the backend. An object store holds the defined objects as dynamic content at runtime and thus provides access to the actual information for front- and backend. The connection to involved backend services is defined by a service model used to call the required backend services. The user interface itself is currently defined via multiple monomodal velocity (http://velocity.apache.org) templates associated with each task. The templates define the actual user interface, and incorporate the dynamic information from the object store. The selection of the active templates is carried out based on the active interaction tasks. The utilization of multiple monomodal interface templates allows forming a multimodal user interface. Interactions received via one of the modalities are interpreted and mapped onto domain object manipulations or task completions. In combination with interaction channels [1] that can be set up to interaction devices on the fly to render and transport the results of the templates, task completions and object manipulations are reflected in all active presentations, which allows the synchronization of the different monomodal UI parts via the underlying model.

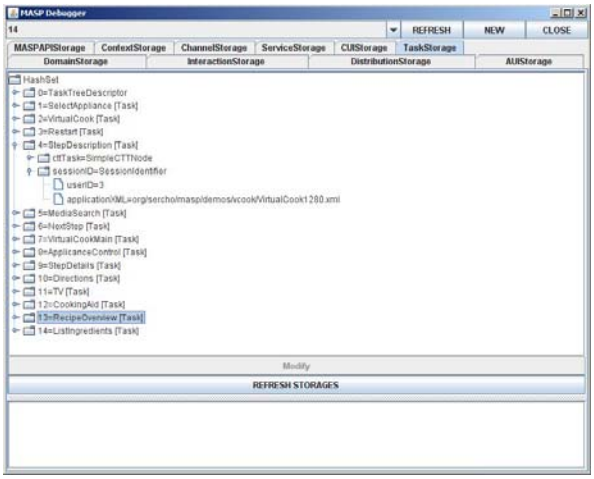

Figure 1: The MASP Debugger

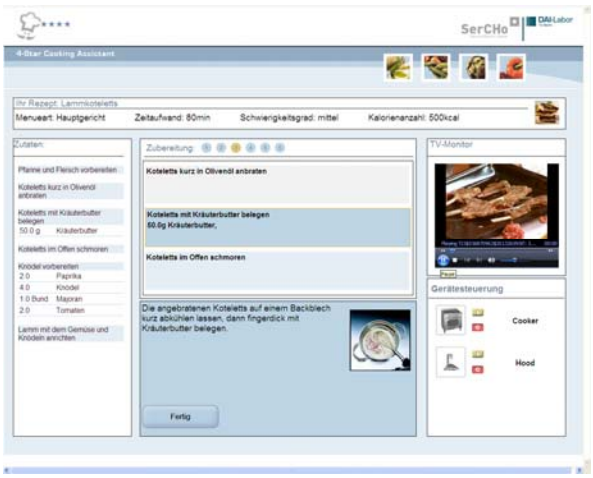

Figure 2: The graphical user interface of the cooking aid

Utilizing models at runtime also allows the inspection of the state of the application stored as dynamic part of the models. Figure 1 shows the debugger that can be used to browse through the actual state of the application. The tool connects to the runtime system and allows to inspect and alter the models for prototyping or direct manipulation of the running application. To evaluate the approach we built an application using the multimodal interaction and distribution capabilities the described approach provides.

3. THE COOKING ASSISTANT

We developed a cooking assistant (CA) (Figure 2) as example application, to evaluate our runtime system. The CA has then also been deployed as part of an ambient living testbed setup at the DAI-Labor at the TU-Berlin as part of the Service Centric Home project (www.sercho.de). The CA is based on three interaction steps. First the user selects a recipe, from the results of a search according to criteria given by the user. Afterwards an interactive dialog queries the user about what ingredients are available. Based on this information a shopping list is generated. Finally the CA guides step by step through the cooking process.

The whole application can be controlled via mouse, keyboard, touchscreen or voice and feedback from the system is provided via a graphical user interface as well as via voice output. The combination of the different modalities is determined based on the availability of the required interaction resources. Thus, the interactive querying of the availability of the ingredients can either be done via voice or via the graphical user interface. However, as the user has to move freely around in the kitchen, using voice interaction seems to be more appropriate in this case. Once the shopping list has been generated, the user can migrate the list to a mobile device using the distribution feature of the MASP. This allows to continue interaction during shopping, by marking the bough ingredients. Once shopping is done, the user indicates that, and seamlessly continues with the cooking assistant. The CA then guides step by step through the cooking process (Figure 2) and the user is able to control kitchen devices (e.g. turn on the oven) and request additional explanations in form of a video for each step. Device and video control as well as navigation between steps are possible via voice or the graphical user interface. Ingredients and step details are presented visually and via voice output. Voice input can be realized either via speaker dependent dictation or via speaker independent recognition. A small chat style interaction application allows text input via dictation or the keyboard, e.g. to realize Wizard of Oz experiments.

The cooking assistant serves as example to demonstrate multimodal interaction based on voice and speech via the MASP. It shows how different channels and modalities can be added and removed on the fly. The shopping list scenario illustrates the capability to distribute the developed user interfaces across multiple devices while keeping the different parts synchronized.

4. ACKNOWLEDGMENTS

We thank the German Federal Ministry of Economics and Technology for supporting our work as part of the Service Centric Home project in the Next Generation Media program.

5. REFERENCES

[1] Blumendorf, M., Feuerstack, S. Albayrak, S. Multimodal user interaction in smart environments: Delivering distributed user interfaces. *European Conference on Ambient Intelligence: Workshop proceedings*, 2007.

[2] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 2003.

[3] Clerckx, T., Vandervelpen, C., Luyten, K., Coninx, K. A task-driven user interface architecture for ambient intelligent environments. In *Proceedings of IUI '06*.

[4] Feuerstack, S., Blumendorf, M. and Albayrak, S. Prototyping of multimodal interactions for smart environments based on task models. *European Conference on Ambient Intelligence: Workshop proceedings*, 2007.

[5] Paternò, F.. *Model-Based Design and Evaluation of Interactive Applications*. Springer 1999.

Hard- / Software Requirements

The proposed demonstration is based on a Java application running inside a Tomcat webcontainer. Additionally Dragon Natural Speaking is used for voice interaction. Voice Output is provided via a Voice Genie Server running externally.

Thus, the hardware requirements are:

- a Java enabled PC running a Tomcat webserver
- windows operating system is preferred
- soundcard, microphone and speakers are required for voice interaction
- an Internet connection that supports a VPN tunnel to a server located at the Technical University of Berlin is required for speech output

The software requirements are:

- Java version 6
- Windows operating system
- Tomcat Webcontainer version 5 or 6
- Dragon Natural Speaking

In case of the acceptance of our proposal we would be happy to bring all the needed equipment in form of a laptop with the complete system installed.