DAI-Labor
TU Berlin

Marco Blumendorf, Sebastian Feuerstack, Sahin Albayrak:

Event-based Synchronization of Model-Based Multimodal User Interfaces

Zuerst erschienen in:

Proc. Of Second International Workshop on Model Driven Development of Advanced User Interfaces, MoDELS 2006, Genova, Italy, 1.-5.10.2006

DAI-Labor, Technische Universität Berlin
Fakultät IV für Elektrotechnik und Informatik
www.dai-labor.de

Event-based Synchronization of Model-Based Multimodal User Interfaces

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

ABSTRACT
Smart environments utilize computers as tools supporting the user in his daily life, moving interaction with computers from a single system to a complex, distributed environment. User interfaces available in this environment need to adapt to the specifics of the various available devices and are distributed across several devices at the same time. A problem arising with distributed user interfaces is the required synchronization of the different parts. In this paper we present an approach allowing the event-based synchronization of distributed user interfaces based on a multi-level user interface model. We also describe a runtime system we created, allowing the execution of model-based user interface descriptions and the distribution of user interfaces across various devices and modalities using channels established between the system and the end devices.

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
Multimodal interaction, user interface model, distributed user interfaces, synchronization, ubiquitous computing, smart environments

1. INTRODUCTION
The ever increasing processing power of current personal computers supports the development of increasingly complex applications. The focus is moved from interacting with the computer to utilizing the computer as a tool supporting users in solving everyday problems. Computer systems increasingly move to the background and change to silent servants ubiquitously available in smart environments. In combination with the emergence of new devices supporting different interaction modalities (pen-based input, voice-, mouse-, touch-, and gesture-based interaction) this offers new interaction possibilities allowing the user to choose the most feasible device for a specific task. The simultaneous availability of these capabilities also allows the combination of multiple devices and modalities, increasing the available communication bandwidth to interact with the computer system. However, the dynamic distribution of user interfaces that is required for this kind of interaction is a task facing several technical problems. The device independent description and the decomposition of user interfaces are currently tackled by several model-based approaches [3][4][10][2], researching for new ways to define user interfaces in a device independent manner. Such a system is required to dynamically adapt to changes in the environment to support flexible human-computer interaction, allowing the user to change, add and remove interaction devices according to the executed task. Distributing user interfaces in such a manner requires the coordination of the different presentations and the resulting input from the user. A mechanism to synchronize the views and update the presentation is needed, as well as a mechanism allowing the interpretation of the user input. The system has to assure that the different views are consistent and provide a usable view of the system.

In this paper we present an approach supporting multimodal human-computer interaction allowing the user to increase interaction capabilities and expressiveness by dynamically combining multiple modalities. The coordination of the different parts of the user interface takes place via event propagation through a multi-level model-based user interface as defined in the Cameleon reference architecture proposed in [2]. An implementation of the approach is described, based on our runtime environment for model-based multimodal user interfaces, supporting event-based coordination, the Multi-Access Service Platform (MASP).

The remainder of this paper is structured as follows. In section 2 we present the related work in this area. Section 3 describes our approach to multi-level event propagation allowing the coordination of distributed user interfaces. Afterwards we describe our implementation of the Multi-Access Service Platform, incorporating a first prototype of the approach. We conclude with a summary and outlook in the final section.

2. RELATED WORK
Common authoring approaches rely on model-based mechanisms such as [8][10][6] and use transformations on different levels of abstraction to generate multi-modal user interfaces. The basis for most approaches is a task tree notation based on the Concurrent

MDDAUI'06, October 2, 2006, Genova, Italy.

Task Tree notation [7]. Most of the current approaches focus on the definition of models for the creation of user interfaces at design time, but there are also ongoing efforts to realize the user interface generation using a model interpreter at runtime [5][6] to dynamically adapt to the interaction capabilities offered by the connected modalities.

Most multimodal approaches we are aware of render the user interface model to a single multimodal final user interface definition like XHMTL+Voice as it is done described in [10] for example. These approaches are limited to single devices, handling internally the synchronization of input and output-modalities. Other frameworks offering comprehensive multi-modal user interfaces such as [1] concentrate on specific environments like cockpit control or on multi-modal interaction with an avatar [9]. These approaches have a strong focus on specific domains and are usually connecting the supported modalities closely together, as all participating modalities are known in advance.

The availability of dynamic environments, providing a combination of devices unknown at design time requires approaches allowing the dynamic derivation of user interfaces at runtime and their distribution and fission based on the available devices [11]. In most cases bridging different technical standards to connect all devices available in smart environments is still a challenge, especially when devices are dynamically selected and a synchronization of the distributed user interface is required.

The various approaches allow the definition of user interfaces that can be delivered to different devices as well as the design of distributed user interfaces. However, it is yet unclear how the dynamic (re-) distribution of user interfaces at runtime and the coordination of the distributed parts can be realized in detail. In the next section, we present our approach to dynamic coordination of distributed multimodal user interfaces.

3. MULTILEVEL EVENT PROPAGATION

Our approach to the synchronization of distributed user interfaces is based on a messaging mechanism, allowing the propagation of events through a multi level user interface model. The model our approach is based on incorporates the following levels: conceptual level, abstract UI, concrete UI, and final UI, as proposed by the cameleon reference framework in [2]. The different levels of the user interface model and the mappings between the levels refine the presentation of the UI when moving from the conceptual model to the final user interface (FUI) and add semantic meaning to the presented elements when moving from the FUI to the conceptual model. Final user interfaces (FUIs) are thereby generated by top-down reification mechanisms, refining the presentation information based on the different abstraction levels of the model. As we focus on smart environments we target the combination of multiple interaction devices to simultaneously access one application. The actual view to the system is thus defined by a set of generated FUIs, distributed across multiple devices with each FUI being adapted to the specific capabilities of the device. This system of distributed FUIs forms a highly dynamic and complex environment that requires the synchronization of the different parts at runtime.

In our approach we realize the required synchronization via the propagation of messages through the defined user interface model. In the same way the reification can be used to derive user

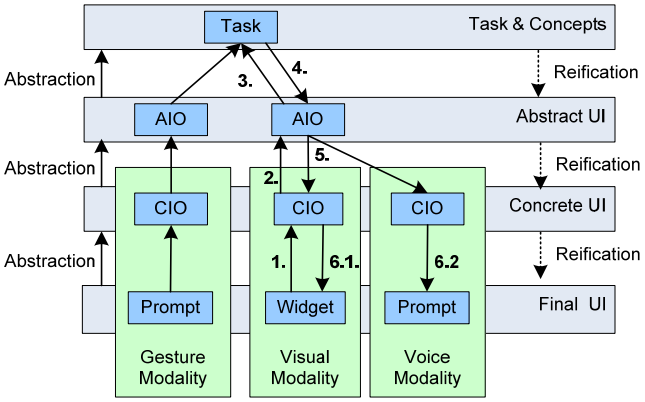

Figure 1: Hierarchical multilevel event propagation using the Cameleon Reference Architecture

interface presentations for specific devices, the abstraction can be used to interpret user input events communicated bottom-up. Input events issued by the FUI are propagated step by step through the user interface levels being semantically enriched, to allow their interpretation on the abstract and conceptual level of the model. In the same way events from the FUI are abstracted it is also possible to use the reification to derive output messages, updating the specific presentation from abstract events resulting from the user input interpretation. The combination of the two mechanisms allows the coordination of the different parts of the distributed user interface based on event propagation mechanisms. The fact that events are either directly interpreted by the specific layer or propagated to the next layer without directly affecting it avoids event conflicts occurring when different FUIs receive conflicting input. This allows recognizing and handling conflicts at the affected layer before lower layers have been altered. Figure 1 depicts the model responsible for the creation of the distributed FUI, which spans a tree across the different levels of abstraction. This entails that the different final user interfaces share a common root node allowing the synchronization of the different representations via the propagation of events through this root node.

As illustrated in Figure 1 an event fired by user interaction (for instance moving the mouse over a widget) is first processed by the final user interface and mapped to a concrete interaction object (CIO). The platform specific “onmouseover” event could thus be transformed to a more abstract focus event on the concrete UI model (1). This abstraction involves looking up the CIO that has been associated to the widget that fired the “onmouseover” event. In our approach each CIO knows its parent abstract interaction object (AIO) on the next abstraction level (whereas the AIO does not know all its derived CIOs). Before the event is propagated to the abstract UI layer, it is abstracted to a “focus” event and associated to an AIO (2). On the next level, the abstract UI processes the event and relates it to the task model of the user interface (3). A specific task receiving the focus, results in a “setfocus” event, propagated the same path backwards, as all final user interfaces displaying the element now have to be notified about the changed focus. During this top-down event propagation (reification) the “setfocus” event issued from the task level is propagated to the derived abstract UIs (4). On the AUI level, the events are related to the involved AIOs and then further propagated to the CUI level (5). Here the events are again mapped to the associated CIOs and interpreted depending on the targeted

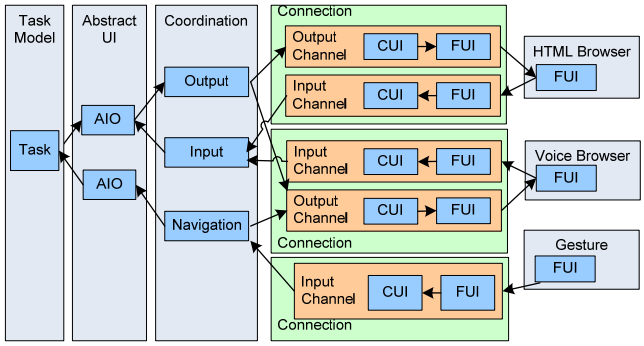

Figure 2: Synchronization via coordination topics of loosely coupled connections in MASP architecture

output modality. Finally the adapted events are delivered to the FUI level (6.1+6.2), where they result in an update of the specific presentation. In a visual modality, the "setfocus" event could result in a highlighting of a widget, whereas in a voice-based modality, the event could result in a speech output.

To evaluate the described event propagation mechanism and the classification of the events for the dynamic coordination of distributed user interfaces we developed a runtime system for a model based user interface. Based on a task tree model the event abstraction and reification mechanisms as well as an event classification are combined to provide multimodal user interfaces, allowing the multimodal usage of web-based application via multiple interaction channels that can be added and removed independently at runtime. In the following section we describe our implementation of the Multi-Access Service Platform, allowing accessing an application described by a user interface model via various channels.

4. THE MULTI-ACCESS SERVICE PLATFORM

The Multi-Access Service Platform (MASP) has been realized as a framework allowing event-based synchronization of distributed user interfaces based on a hierarchical user interface model. The delivery of final user interfaces and messages is realized via connections to devices supporting two-way client-server communication (Figure 2).

Connections established between the MASP and any interaction device accessing the MASP realize event-based, two-way client-server communication, by abstracting from the underlying communication mechanism (i.e. HTTP). In our understanding a connection is a way to describe the communication with a specific device, acting as a container combining different *communication channels* to abstract from the device specifics. A communication channel is part of a connection and responsible for the one-way communication of events. We distinguish between input channels, providing user input events to the system and output channels allowing the manipulation of the FUI via output events. Each channel provides eventing capabilities and is connected to different topics, allowing the classification of events.

Interaction with the system takes place via events fired by the user interface through the channels. These events are processed by our system and delivered to the affected parts of the user

Figure 3: The graphical user interface of the cooking aid

interface model. To provide a general abstraction layer from the multiple events that can be fired by the final user interface (i.e. onmouseover, onmouseout, onclick, onblur, etc in HTML) we introduce three types of interaction events: *focus*, *input* and *output*. *Focus events* have a navigational nature, covering events that do not change the status of the system, but the status of the current view of the system. *Input events* have an interaction nature, covering selection and text input triggered by the user. *Output events* are events issued by the system to adapt the presentation of the user interface to the current status of the system. They allow to synchronize FUIs when the presentation changes. A FUI presented on an end-device can issue *focus* or *input events*, whenever a user interaction occurs and receive output events when the presentation has to be updated. The mapping of FUI specific events to a supported interaction event is provided by the channel, managing the communication with the specific FUI. The interaction channel thus provides a device and modality abstraction, introducing a common interaction mechanism. In our implementation we are using Java Messaging System (JMS)-based messaging, allowing the flexible distribution of messages to the affected system components. Events received through an interaction channel are propagated to the backend model through a number of topics, allowing the classification of the received events and their appropriate distribution.

In addition to the interaction events we defined additional events on the level of the task model (task done, task disabled, task enabled) to communicate changes on the task level. We define that specific input events can be mapped to task done events by the abstract user interface. Task enabled and disabled events are mapped to output events, taking care that the specific task presentations are shown or hidden from the specific FUIs. In our implementation the task model is defined using the Concurrent Task Tree notation [7], interpreted at runtime.

Using connections the runtime system can be dynamically connected to various devices by setting up the required channels. Once a channel is set up, the system can render a final user interface for the channel and deliver it to the device. A user interface can be distributed across multiple devices and modalities when multiple channels are available. A mechanism of sending updates to the presented user interfaces via output events allows the redistribution and adaptation of user interfaces when new device enter or leave the interaction environment.

5. The Virtual Cook

As an example, demonstrating the usability of our framework, we created a Virtual Cook, presenting a cooking aid, showing the required steps to support the user when preparing a meal. Figure 3 shows the graphical user interface of the virtual cook. As a person usually doesn't have the hands free for using mouse and keyboard during cooking, we equipped the Virtual Cook with a voice based interface, which can support the control of the visual output. Besides the possibility to dynamically add and remove a voice channel when using the application we also added support for a gesture recognition channel. The voice channel is realized via SIP-based communication, allowing a loose coupling of the voice channel. To connect the gesture channel we created an interface defining five gestures for navigation in the cooking aid user interface (back, forward, up, down and step done). This interface can be delivered to a gesture recognition device we build ourselves, extending the possible interaction modalities via gesture-based interaction.

Our implementation of the virtual cook application using the MASP framework to realize an enhanced multimodal user interface allows the distribution of the user interface across various modalities based on the availability of the devices. The connection abstraction allows us to dynamically add and remove devices from the environment which results in interaction modalities being added/removed to/from the application.

6. CONCLUSION

In this paper we introduced the event-based synchronization of distributed user interfaces, based on a hierarchical user interface model, defining the different aspects of the UI on multiple levels of abstraction. The framework we presented allows processing of user input events and synchronization of dynamically distributed user interfaces. A connection-based communication mechanism combining multiple channels can be used to communicate with the user via multiple modalities.

However, in our work we focused on the extension of a primary modality with additional redundant interaction capabilities. The extension of the approach to support any mixture of modalities as well as the usage of complementary user interfaces requires more research considering the elimination of ambiguous events and the fusion of multipart events.

We also did not set a strong focus on the rendering of user interfaces for the different modalities from one common model, but rather annotated a task tree with user interfaces for the different supported modalities.

In the future, we also want to support more gesture and voice commands and a more flexible definition of the user interface model, considering the different interaction styles of the modalities in a more appropriate way.

The presented approach provides an event-based mechanism incorporating the multi-level structure of model-based user interfaces to coordinate distributed user interfaces. However, the presented implementation is still not complete and can be extended towards better support for the new possibilities provided by multimodal human-computer interaction in smart environments.

7. 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.

8. REFERENCES

[1] Bouchet, J.; Nigay, L. & Ganille, T. (2004),ICARE software components for rapidly developing multimodal interfaces, *in* 'ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces', ACM Press, New York, NY, USA, pp. 251-258.

[2] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L. and Vanderdonckt, J. A Unifying Reference Framework for Multi-Target User Interfaces. Interacting with Computers 15, 3 (2003), 289–308.

[3] Coninx, K.; Luyten, K.; Vandervelpen, C.; den Bergh, J.V. & Creemers, B. (2003),Dygimes: Dynamically Generating Interfaces for Mobile Computing Devices and Embedded Systems., *in* 'Mobile HCI', pp. 256-270.

[4] Eisenstein, J.; Vanderdonckt, J. & Puerta, A.R. (2001), Applying model-based techniques to the development of UIs for mobile computers, *in* 'Intelligent User Interfaces', pp. 69-76.

[5] Klug, T. & Kangasharju, J. (2005), Executable Task Models, *in* 'Proceedings of TAMODIA 2005', ACM Press, Gdansk, Poland, pp. 119-122.

[6] Mori, G.; Paterno;, F. & Santoro, C. (2003), Tool support for designing nomadic applications, *in* 'IUI '03: Proceedings of the 8th international conference on Intelligent user interfaces', ACM Press, New York, NY, USA, pp. 141--148.

[7] Paterno, F (1999), Model-based Design and Evaluation of Interactive Applications. Springer Verlag. Berlin 1999.

[8] Paterno, F. & Giammarino, F. (2006), Authoring interfaces with combined use of graphics and voice for both stationary and mobile devices, *in* 'AVI '06: Proceedings of the working conference on Advanced visual interfaces', ACM Press, New York, NY, USA, pp. 329-335.

[9] Reithinger, N.; Alexandersson, J.; Becker, T.; Blocher, A.; Engel, R.; Löckelt, M.; Müller, J.; Pfleger, N.; Poller, P.; Streit, M. & Tschernomas, V. (2003), SmartKom: adaptive and flexible multimodal access to multiple applications, *in* 'ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces', ACM Press, New York, NY, USA, pp. 101-108.

[10] Stanciulescu, A.; Limbourg, Q.; Vanderdonckt, J.; Michotte, B. & Montero, F. (2005), A transformational approach for multimodal web user interfaces based on UsiXML, *in* 'ICMI '05: Proceedings of the 7th international conference on Multimodal interfaces', ACM Press, New York, NY, USA, pp. 259-266.

[11] Vandervelpen, C. and Coninx, K. Towards model-based design support for distributed user interfaces. In *Proceedings of the Third Nordic Conference on Human-Computer interaction* (Tampere, Finland, October 23 - 27, 2004). NordiCHI '04, vol. 82. ACM Press, New York, NY, 61-70.