Multimodal User Interaction in Smart Environments: Delivering Distributed User Interfaces

Marco Blumendorf, Sebastian Feuerstack, Sahin Albayrak

DAI-Labor, Technische Universität Berlin
Ernst-Reuter Platz 7, D-10587 Berlin, Germany
{Marco.Blumendorf, Sebastian.Feuerstack, Sahin.Albayrak}@dai-labor.de

Abstract. The ongoing utilization of computer technologies in all areas of life leads to the development of smart environments comprising numerous networked devices and resources. Interacting in and with such environments requires new interaction paradigms, abstracting from single interaction devices to utilize the environment as interaction space. Using a networked set of interaction resources allows supporting multiple modalities and new interaction techniques, but also requires the consideration of the set of devices and the adaptation to this set at runtime. While the generation of user interfaces based on UI models, although still challenging, has been widely researched, the runtime processing and delivery of the derivable user interfaces has gained less attention. Delivering distributed user interfaces while maintaining their interdependencies and keeping them synchronized is not a trivial problem. In this paper we present an approach to realize a runtime environment, capable of distributing user interfaces to a varying set of devices to support multimodal interaction based on a user interface model and the management of interaction resources.

Keywords: multimodal distributed user interfaces, model-based user interfaces, human-computer interaction, smart environments

1 Introduction

The integration of multiple computer-based appliances in people's homes and the convergence of home electronic and computer systems lead to networks of powerful devices, comprising multiple connected Interaction Resources (IRs) [7] (i.e. screen, speaker, microphone or mouse) that provide access to services and applications. Microphones, cameras, accelerometers and specialized instruments as well as e.g. screens and speakers make multimodal user input perceivable for the system and present multimedial system output to the user. While the generation of user interfaces based on UI models has been widely researched, the runtime interpretation and delivery of the derived user interfaces for such highly distributed environments has gained less attention yet.

Problems arising from utilizing distributed interaction in smart environments are the development of multimodal user interfaces distributed across various interaction

resources as well as the delivery of the parts of the user interface and their synchronization across loosely coupled collections of interaction resources in smart environments.

This paper presents an approach, addressing these challenges by the development of a runtime system for the interpretation of model-based multimodal user interfaces. The system combines an interpretation engine for user interface models with a channel-based user interface delivery mechanism. Based on previous work presented in [2] we develop a definition of interaction channels and describe the implementation of the approach, focussing on the integration into our model-based runtime system. After describing the state of the art in the next section, we introduce the idea of interaction channels and describe their utilization for the delivery of user interfaces. In section 5 we give a brief overview of the implementation of our model-based runtime system followed by the conclusion and outlook.

2 Related Work

The basic process of interaction between human and computer can be described as the bidirectional exchange of information between the (digital) computer and the (analog) human user - both with an internal representation - via a physical medium (light, sound or kinetic energy) [9]. To access the physical medium, an *interaction resource* is required allowing perceiving or affecting the medium. [7] defines an interaction resource as “atomic input or output channel” to the user, thus addressing only one modality. Such an *atomic channel* can be described from two points of view: From the human perspective it represents the human sense (vision, hearing, smell, taste, and touch) and motory system used to perceive or affect the physical medium. From the system perspective it connects the interaction resource and thus the physical medium to the system internals.

The term “*channel”* has thereby been used in different contexts and there is no common understanding of the term yet. While [1] uses the term “communication channel” to refer to mouse, keyboard or voice as input and visual, audio or haptic as output, [3, 4] uses the term “presentation channel” to classify the input and output format (HTML/WML). Braun and Mülhäuser utilize the idea of multiple channels to connect several devices grouped in a federation to an interactive system [5]. Nigay et al. [10] define a *communication channel* as the “temporal, virtual, or physical link that makes the exchange of information possible between communicating entities” and based on this multimodality as the "capacity of the system to communicate with a user along different types of communication channels and to extract and convey meaning automatically".

3 Interaction Channels

In this work, we want to introduce the term “Interaction Channel” to identify the complete chain, mediating between the system and the human user (Figure 1). Considering that user and system both have an internal representation of their

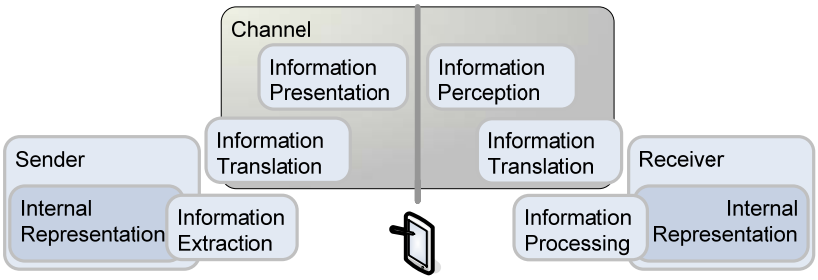

Figure 1: Channels mediate between the internal representation of the system and the user.

knowledge there is an urgent need to mediate and translate between the two. While the systems internal representation is defined by an application model allowing the derivation of resource specific user interfaces, the user maintains a mental model in her head and perceives the state of the system via her senses, mediated by interaction resources. Each channel can be split in a sender- and a receiver-part, with an interaction resource connecting the two sides. Which side is system or user switches depending on the direction of the information flow (input vs. output). Information flow between the two sides requires the following steps: First the sender extracts the information to convey and decides how to communicate by selecting an appropriate channel. Afterwards the information is translated into a channel-specific format and finally presented via an interaction resource. The presented information is perceived by the receiver and translated into an understandable format. The translated information is processed and incorporated into the receiver's internal representation. Applying this approach to the idea of a runtime system creating, delivering and synchronizing user interfaces by interpreting a UI model leads to the need for an implementation of the channel concept on the system side.

4 Delivering User Interfaces

Focusing on the system view of an interaction channel, the channel provides access to the interaction resource properties and also takes care of the delivery of the derived UI and its synchronization with the state of the system. Delivery of UI parts to different interaction resources and the synchronization of the different parts is especially important when considering distributed UIs in smart environments. Utilizing the concept of interaction channels on the system side allows the free combination of multiple resources and their utilization in new and unconventional ways (e.g. the direct control of a large display via a PDA or the creation of a GUI controllable via voice input and mouse gestures). Dynamically assembling sets of resources used for interaction allows the user to combine various input and output modalities according to her specific needs during interaction. Similar to assembling a device like e.g. a PDA, the user first picks the different resources e.g. display, touchpanel, keyboard and stylus pen to define the environment. Afterwards interaction channels established to these resources allow the communication and make the resources available for the use with the interactive system. The independent

addressing of interaction resources and a separation of input and output resources becomes especially important in such a scenario and allows addressing multiple modalities through distribution across different devices.

To enable the dynamic addressing of interaction resources, we utilize a user interface model [6, 8, 12], defining the various aspects of the user interface on different levels of abstraction, following the ideas of the Cameleon Reference Framework [6]. Based on this user interface model our runtime system derives a partial user interface for a specific interaction resource, e.g. an output UI for a screen or a VoiceXML UI for voice input. The derived partial UI describes the final user interface for the interaction resource and also provides support to map updates to the state of the system to UI updates. In case of a partial input user interface addressing an input interaction resource the UI also provides information about how to map the input received from the interaction resource to the format of the internal representation of the system. An interaction channel is able to receive such a partial user interface and deliver it to the interaction resource. In our approach the channel maintains a connection to each interaction resource, providing the basic mechanisms required to push information like a UI to present to the resource. This connection enables the interaction channel to receive the partial user interface, derive the final user interface and deliver that user interface to the connected resource. Once the user interface has been delivered, the channel is responsible for the delivery of updates triggered by changes to the state of the application to the user interface and for receiving and interpreting user input. The propagation of state changes allows the synchronization of multiple user interfaces according to the state of the internal UI model of the system. As soon as the model is changed an update message is created and delivered to the channel, altering the partial UI the channel maintains. This alteration is then also communicated to the final user interface and also alters its presentation (or, in case of an input user interface, the processable inputs for example). Input events received from an interaction resource are interpreted by the interaction channel based on the partial input UI it maintains and mapped to the internal format of the system. This allows the manipulation of the user interface model by such input events, which then again can trigger update events altering the final user interfaces.

5 The Multi-Access Service Platform

Our implementation of the concepts described above is based on the runtime interpretation of a user interface model. Different other approaches follow the idea of utilizing the user interface model at runtime [8] or utilizing similar concepts as our interaction channels to deliver user interfaces [11]. In contrast to other approaches we focus on the utilization of models at runtime to derive partial user interfaces, which can be distributed to various interaction resources connected via interaction channels. The combination of model and delivery allows a stronger integration of model and final UI and the synchronization of distributed UIs based on the underlying common user interface model.

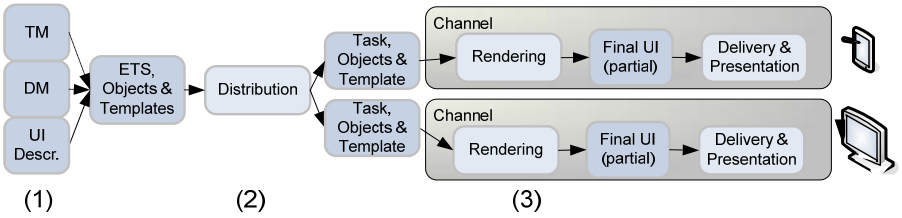

Figure 2: Basic MASP Architecture and information flow during UI derivation and delivery.

In our current implementation, the task model, described in the CTT notation [13], defines the workflow of the application and a domain model defines the objects manipulated by the defined tasks and thus the internal state of the application. Each leaf task references objects defined in the domain model as well as a user interface description allowing the creation of partial user interfaces for each task. User interface descriptions are currently realized as velocity templates[1] allowing the creation of user interfaces in various output formats, based on the objects referenced in the related task. Figure 2 (1) shows the parts of the model.

During the user interface creation process in a first step the Enabled Task Set (ETS) is derived from the task model and the related objects and user interface descriptions are loaded. According to the set of available interaction channels the set of tasks is distributed to the channels. [Figure 2 (2)] For our current applications we are supporting multiple templates to support the multiple types of interaction resources and initially deliver all output tasks to all output channels that templates have been defined for and all input tasks to all input channels respectively. However, we also support the manipulation of the distribution by the user by sending single tasks to different channels and implemented a simple algorithm to calculate the distribution of related tasks based on the "distance" of two tasks, the processed objects and the type of the tasks (input or output task). The distance thereby denotes the number of parent tasks that have to be passed when moving through the hierarchy from one task to the other. The closer the tasks, the more likely they are directly related and should be presented together, especially if they also work on the same objects. However, this approach requires more flexible UI templates allowing the necessary adaptation of the user interface when altering the distribution.

Figure 2 (3) shows the connected channels. Whenever a channel receives the request to deliver a partial user interface to the connected resource, the velocity templates are rendered, producing final user interfaces based on the passed objects defined in the related task. The result of the rendering process is then delivered to the interaction resource, presenting the UI to the user. Interaction resources can thereby be manually registered, by calling a URL or be discovered using UPnP. Technologies like CC/PP also allow the description of the resource specifics, which is currently not directly supported in our implementation.

Interaction with this system takes place in two ways. On the one hand changes to the system model are communicated as UI updates via the channel and alter the presented UI. The annotation of the related objects in the task tree allows the relation of objects to templates as well as the implementation of an observing mechanism for

[1] http://velocity.apache.org/

the domain model, ensuring that each UI element, referencing a domain object is updated as soon as the domain object changes. On the other hand, user input is processed via interaction mappings. Interaction mappings for each interaction task describe the relation of the receivable input to the user interface model. In the current implementation these user interaction is communicated via the event types described in [2] (FocusEvent, SelectionEvent, InputEvent). However, to provide a meaning to these events, and enable them to alter the underlying application model, any of the above events is mapped to MASP internal events according to the defined interaction mapping. MASP internal events currently comprise:

- DSWrite, allows the direct manipulation of objects stored in the domain model. A DSWrite event therefore contains a query to the object, which has the form ObjectID(.FieldID)*.

- TaskDone, allows indicating that the user has finished performing a task, which results in marking a task as done.

Based on a given partial UI description an interaction channel is able to create a mapping between the interaction events and the model manipulation events.

In our implementation we currently support the creation of HTML and VoiceXML code for *output user interfaces* and HTML, VoiceXML and XML-based gesture descriptions for *input user interfaces*. In both cases (input and output) the execution of the template creates a code snippet (a tag in case of HTML and a <form> tag in case of VoiceXML). For HTML, using Javascript, initially deployed to the browser when the channel is established, allows adding tags to the displayed web page initially or replacing existing whenever information in the model changes. The layout is in this case defined by Cascading Style Sheets (css) allowing the definition of the properties of the tags, e.g. the position. The position can either be determined by the interface developer or be calculated by a layouting algorithm, which allows the dynamic positioning of the elements when distributing the UI. An input channel allows the processing of input by delivering Javascript code executed by the browser that creates the related events. Currently the relation of the input to the output presented e.g. “pressing a button” is done by the browser. In a next step we move that processing to the server, to be able to e.g. relate mouse coordinates to button positions on server side, which allows more advanced user interaction like simple mouse gestures or the increasing of the active area of a button without increasing the button size.

The described implementation allows the creation of multimodal user interfaces distributed across multiple interaction resources. Resources currently supported are screens, mouse and keyboard – in this case we use a web browser as an integration platform providing access to the resources via Javascript functions – as well as voice input and output – in this case we use a VoiceGenie voice server – and gesture based input. Gesture based input is realized via a small device we build, that allows the interpretation of simple gestures detected by an accelerometer.

6 Conclusion and Future Work

This paper introduced the Multi-Access Service Platform and an approach to support the delivery of user interfaces to sets of interaction resources to allow the flexible combination of multiple diverse interaction resources. This approach on the one hand allows the creation of multimodal user interfaces for smart environments and on the other hand provides the basis for more advanced user interfaces, capable of dynamically addressing the users' needs. However, during our work we discovered that the dynamic incorporation of multiple interaction resources requires strong user interface models allowing the derivation of multiple different user interfaces. Experimenting with user interfaces separating input and output also made clear, that there is a strong relation between input and output. While the output describes application specific information it also provides the reference point for the user input. This makes it necessary to on the one hand guide the user by communicating input capabilities, provide help systems and supportive output like e.g. a screen keyboard for free text entry via mouse and on the other hand also provide direct input feedback like e.g. the mouse pointer, whenever a user provides (partial) input.

Working on the MASP and several prototypes utilizing the implemented features, it also turned out that the anticipated very flexible interaction in smart environments requires new interaction technologies as well as new paradigms for human-computer interaction. Shifting the human attention away from a single device, towards a whole system of connected interaction resources allows a much more flexible interaction and the utilization of various instruments other than mouse and keyboard to address special needs. However, the creation of such flexible systems requires highly flexible user interfaces, adapting to the users need and environment. This requires a focus shift away from designing user interfaces towards the design of interaction instead. Interaction with smart environments requires interaction concepts, patterns and methodologies not widely researched yet. Our hope is to be able to formalize such interaction concepts into widgets and interaction pattern allowing the flexible creation of easy to understand interaction and the automatic adaptation to the device specifics in the future.

The work described in this paper is part of the Service Centric Home[2] project sponsored by the German Federal Ministry of Economics and Technology. The MASP and a Virtual Cook application developed using the technology is deployed as part of the Ambient Living Testbed which has been set up as part of the project as well. The Ambient Living Testbed consists of a kitchen a living room and an office and provides infrastructure for the development and test of smart home services and the realization of usability studies. Earlier studies with a basic prototype of the system already showed positive feedback for multimodal interaction in smart home environments and underlined the need for the integration of different interaction devices in smart home environments. Currently we are developing several prototypes to integrate multiple interaction resources and allow more flexible interaction in our smart environment. We are very positive that the following experiments will again show positive feedback for the developments.

[2] www.sercho.de

7 References

[1] Abowd, G., Coutaz, J., Nigay, L. Structuring the space of interactive system properties. In *Engineering for Human-Computer Interaction*, volume A-18 of *IFIP Transactions*, pages 113–129. 1992.

[2] Blumendorf, M., Feuerstack, S., Albayrak, S. Event-based synchronization of model-based multimodal user interfaces. In *LNCS MoDELS'06 workshop proceedings: MDDAUI 2006*.

[3] Book, M., Gruhn, V. Modeling web-based dialog flows for automatic dialog control. In *ASE '04: Proceedings of the 19th IEEE international conference on Automated software engineering*, pages 100–109, 2004.

[4] Book, M., Gruhn, V., Lehmann, M. Automatic dialog mask generation for device-independent web applications. In *ICWE '06: Proceedings of the 6th international conference on Web engineering*, 2006.

[5] Braun, E., Mühlhauser, M. Automatically generating user interfaces for device federations. In *ISM '05: Proceedings of the Seventh IEEE International Symposium on Multimedia*, 2005.

[6] Calvary, G., Coutaz, J., Thevenin, D., Limbourg, Q., Bouillon, L., Vanderdonckt, J. A unifying reference framework for multi-target user interfaces. *Interacting with Computers*, 15(3):289–308, 2003.

[7] Clerckx, T., Vandervelpen, C., Coninx, K. Task-based design and runtime support for multimodal user interface distribution. In *Proceedings of Engineering Interactive Systems 2007*.

[8] Coninx, K., Luyten, K., Vandervelpen, C., Van den Bergh, J., Creemers, B.. Dygimes: Dynamically generating interfaces for mobile computing devices and embedded systems. In Luca Chittaro, editor, *Mobile HCI*, volume 2795 of *Lecture Notes in Computer Science*, 2003.

[9] Coomans, M. and Timmermans, H. Towards a taxonomy of virtual reality user interfaces. In *IV '97: Proceedings of the IEEE Conference on Information Visualisation*, 1997.

[10] Nigay L. Coutaz, J., Salber, D. The msm framework: A design space for multi-sensori-motor systems. In *EWHCI*, volume 753 of *Lecture Notes in Computer Science*, 1993.

[11] Elting, C., Rapp, S., Möhler, G., Strube, M. Architecture and implementation of multimodal plug and play. In *ICMI '03: Proceedings of the 5th international conference on Multimodal interfaces*, 2003.

[12] Mori, G., Paternò, F., Santoro, C.. Design and development of multidevice user interfaces through multiple logical descriptions. *IEEE Trans. Softw. Eng.*, 30(8):507–520, 2004.

[13] Paternò, F. *Model-Based Design and Evaluation of Interactive Applications*. Applied Computing. Springer-Verlag, 1999.