Device-Spanning Multimodal User Interfaces

Elmar Braun, Andreas Hartl
Telecooperation Group
Department of Computer Science
Darmstadt University of Technology
Alexanderstr. 6, 64283 Darmstadt, Germany
{**elmar, andreas**}**@tk.informatik.tu-darmstadt.de**

ABSTRACT
Despite the large variety of mobile devices available today, none of them is without flaws: small devices like cell phones are very limited regarding interaction, while larger devices like laptops lack true mobility. We are investigating how both mobility and rich interaction can be achieved by federating small mobile devices with other interactive devices on demand. However, this raises the question how user interfaces can be authored if the target device is not static and known, but rather a changing set depending on user context. We approach this using device independent widgets, which are mapped to device specific widgets at runtime.

1. INTRODUCTION
Mobile devices have continuously become more powerful and cheaper over the last years. However, they still lack flexibility as most of them are designed to be used stand-alone, and offer only limited interaction means. We present an architecture which integrates the multitude of mobile devices with each other to augment the users' experience with them.

While we expect most of such mobile devices can be shared and accessed as needed, we also think that there still will be a personally owned device in such a world for security and privacy reasons. Such a device should be unobtrusive and as small as possible. Our group has designed a headset with voice based interaction and communication abilities. Other devices may be associated to this *Talking Assistant* (TA) [2] as needed, and may also bring in other means of interaction such as graphical user interfaces.

1.1 Example Scenario
What is the benefit of a voice based assistant, and of associating other interactive devices with it on the fly? We show this in a scenario which describes how a user named Alice orders a pizza online while hurrying out of her office. Alice notices that she is hungry exactly as she locks her office behind her. She considers turning back and ordering from her desktop PC, but she is wearing her TA and decides to access the pizza order application by voice menu instead. Alice instructs the TA to access this application and orders a bottle of soft drink. The next step is choosing a pizza topping. Before making a choice, Alice would like to see the menu. But how should the menu be presented to her? Speech synthesis is unsuitable for reading such a long list. Luckily, the building has public displays installed in the hallways. Rather than turning back and restarting the order on a better suited device with a larger display, Alice simply stops in front of one of the displays. The TA notes that it is able to associate with a screen, and relays this information to the order application, which decides to show its menu there. After scanning the menu, Alice simply says ”sixty-five, large” into her TA, and resumes walking. The remaining steps of the order are presented to her as a pure voice menu again.

2. AUTHORING ADAPTIVE APPLICATIONS
For authoring applications that make use of the TA and its associated devices, we are going to enhance the idea of widgets as simple elements of interaction [1]. Widgets currently are considered merely as graphical elements; there is no such high-level approach for other modalities. In our broader approach we think of them as objects that represent a generalized, general purpose solution towards interaction between users and applications.

The framework for programming the TA will provide basic device independent *logical widgets* which developers may use to create their multimodal applications. These are comparable to the *atomic interactors* of XWeb [3]. XWeb features a browser-like approach to data exchange, whereas we provide an event system like traditional GUI widget toolkits. Our widget system is extensible: It is possible to extend existing widgets, to create entirely new ones or to combine widgets to more powerful ones. While further research will be done in this area, we have so far identified the following as some of the basic widget classes to be provided by the system:

- free form text input; output of simple and structured text
- yes/no input for boolean elements
- select one of several mutually exclusive elements
- input and output of date and time
- a grouping element combining several widgets

3. MAPPING WIDGETS TO A DEVICE
Application developers define the user interface of their applications by creating a tree of logical widgets. These logical widgets are purely abstract representations of the user interface with no association to a specific device. In a two step process, these logical widgets are mapped to a device.

First, the mapping subsystem creates *physical widgets* out of logical widgets. Logical widgets are mere data objects representing only the kind of interaction. Their physical counterparts contain methods to render themselves onto a specific modality and/or device. As a result, physical widgets are device dependent. The mapping subsystem utilizes context metadata such as the device used, its primary interaction method (graphics based or voice based) and additional information about the capabilities of the interface (e.g. the voice recognizer used, window metrics, etc.).

Physical widgets are registered to the mapping subsystem with the information what logical widget they map to, what modality they implement, and what constraints they have. Several physical widgets may be registered for one logical widget, e.g. for different modalities or for different implementations of one modality. At runtime, the mapping subsystem searches for the physical widget which best fits the device and chooses it to substitute the logical widget.

In the second step, the physical widgets render themselves onto the user interface. How this is done is modality specific. For voice based interaction, this could involve using text-to-speech for doing the output and generating context free grammars for specifying the input. The equivalent physical widget for GUIs may just call the appropriate element of the operating system's widget toolkit.

4. ASSOCIATION AND MULTIPLE DEVICES

A user interface can obviously not exceed the limitations of the device it runs on. When mapping an interface to a small mobile target device, it may allow basic interaction in the absence of a better terminal. However, mobile devices are not always used in isolation. Often, the surrounding infrastructure could provide additional means of interaction. We intend to dynamically associate mobile devices with devices from the infrastructure in order to overcome their limitations regarding interaction. Since the number of possible combinations of devices is rather large, hand-coding a specialized UI for each combination is infeasible. The mapping subsystem will provide a scheme to render an interface on a *federation* of multiple devices. This concept has so far only been considered for playback of multimedia content [4].

Before such a federation can be established, one must detect that a device is within the user's range and that the device can be associated. We currently use two methods for detecting possible association between users and devices. One is the TA, which determines its wearer's head position (using two cameras tracking an infrared beacon on the TA) and gaze direction. The other consists of tags on each device, which transmit their ID using short range infrared, and badges on each user, which receive a tags' ID if the user is standing in front of the tagged device, and relay them to the network. The advantage of the latter solution is the low cost.

Mapping a user interface to span multiple devices introduces a number of novel problems:

- When adapting for a single device, there is no choice regarding which device to present a widget on. If several devices are available, the mapping needs decide how to distribute widgets to devices, factoring in usability and device characteristics.

- While there are some dynamic device characteristics (e.g. battery status), most characteristics of a single device are fixed. If users move in and out of range of associated devices at runtime, the virtual target device of the mapping changes drastically at runtime, making mapping the UI for multiple devices a much more dynamic process.

- Despite constantly changing context, the mapping should not present the user with a constantly changing interface. This would inhibit usability since the user would have no chance to become accustomed to the UI. Therefore a history of how a UI was presented to a user before needs to be considered as an additional form of context.

- In the case of a single device, each widget is rendered exactly once. When using a federation of devices, it can make sense to render an element more than once, e.g. in different modalities to achieve multimodality, or on different devices to create some form of remote control.

We are investigating several mapping methods that take these criteria into account. Currently we are building a test bed that allows us to create distributed UIs, and to automatically send components of these to different devices in the room infrastructure. This allows us to experiment with distributed UIs, and to try out mapping algorithms for the distributed case. In future work it will be used for user studies.

5. CONCLUSION

We have presented a way to create multimodal applications whose user interface may span across several devices. The approach is based on a generalized concept of widgets as interaction elements. We have developed a mapping subsystem that determines the appropriate mapping of a logical widget at runtime based on the target device. By mapping at runtime we can support several modalities concurrently.

We have shown how several devices may be integrated into federations in order to define the target for device-spanning user interfaces. While the mapping subsystem is designed to cope with several modalities, distributed user interfaces pose additional challenges to the mapping which we have identified and are currently working on.

REFERENCES

[1] A. Hartl. A Widget Based Approach for Creating Voice Applications. In *Proceedings of MobileHCI*, Udine, Italy, 2003. to appear.

[2] M. Mühlhäuser and E. Aitenbichler. The Talking Assistant Headset: A Novel Terminal for Ubiquitous Computing. In *Microsoft Summer Research Workshop, Cambridge*, Sep 2002.

[3] D. R. Olsen, S. Jefferies, T. Nielsen, W. Moyes, and P. Fredrickson. Cross-modal interaction using XWeb. In *Proceedings of the 13th annual ACM symposium on User Interface Software and Technology*, pages 191–200, San Diego, USA, 2000. ACM Press.

[4] T. Pham, G. Schneider, and S. Goose. A Situated Computing Framework for Mobile and Ubiquitous Multimedia Access Using Small Screen and Composite Devices. In *Proceedings of the 8th ACM international conference on Multimedia*, pages 323–331, Marina del Rey, USA, 2000. ACM Press.