

The Ubiquitous Interactor – Universal Access to Mobile Services

Stina Nylander & Markus Bylund

Swedish Institute of Computer Science
Lägerhyddsvägen 18, 75237 UPPSALA, SWEDEN
{stina.nylander, markus.bylund}@sics.se

Abstract

The Ubiquitous Interactor is a system for universal access to mobile services, where the user-service interaction is the common denominator between different devices. By combining interaction with device specific presentation information, tailored user interfaces for different devices can be created. A calendar service has been developed as a proof of concept.

1 Introduction

Today, users have a wide range of devices at their disposal for accomplishing different tasks: desktop computers, laptop computers, wall-sized screens, PDAs and cellular phones. Unfortunately, users cannot freely combine devices and services. Many services are accessible from single devices, and those who are accessible from different devices mainly use two approaches. Either they present themselves the same way to all devices, or they have many different versions. Both approaches have drawbacks: the same presentation for all devices either gets very simple, or excludes thinner devices, and is not utilizing device specific features. Creating new versions of services for each device is very costly in development and maintenance work.

The Ubiquitous Interactor (UBI) is a system that allows service providers to, once and for all, design services that can be accessed via an open set of user interfaces (e.g. web UI or GUI) on many types of devices (e.g. desktop computer, PDA, and cellular phone). It also allows adaptation of the user interface to particular device or user interface types. This adaptation can be applied, and even re-applied, at any time in order to cater for changes in users' needs. As such, UBI gives service providers the means for easy implementation and maintenance of services that adapt their user interfaces to individuals and groups of users. This is particularly suitable in a mobile setting where users' needs, access to devices, and ability to interact with services is known to vary greatly. With UBI, users can choose a device that is suitable for accessing a service in a certain situation, and get a user interface adapted to that device.

UBI separates adaptations from the interaction with services. The interaction is kept the same over all devices, while the adaptations are specific both to a service and to a device or a type of user interface. This means that the same version of a service can be accessed from many different devices, and that new adaptations can be created at any time, even by third party provisioners.

2 Interaction as abstraction level

The Ubiquitous Interactor separates interaction from adaptation. The interaction is kept the same between a service and all devices used for access. This way, services do not need to keep track of

which device is used for access. To make this work, a suitable level of abstraction needs to be identified. In UBI, we are working with the *user-service interaction* as level of abstraction. User-service interaction is defined as *actions that services present to users, as well as performed user actions, described in a modality independent way*. According to this definition, examples of interaction are: presenting information to the user (text, images or sound), input data to the system, presenting alternatives for the user to choose from, or performing a choice (by clicking in a menu, pressing a telephone button or speaking). These actions do not change over devices or modality, even though the presentation of them can change considerably.

The interaction is encoded in *interaction acts*, to form a general description containing no information about presentation. The general description can be complemented with an optional *customization form* containing device and modality specific information to generate a user interface tailored both to the current service and to the device used for access. This means that services expressing themselves in interaction acts can be accessed from any device that interprets them, be it mainstream cellular phones or custom made assistive technology. If a customization form is created, by the service provider or a third party, a tailored user interface can be generated.

2.1 Interaction acts

An interaction act is an abstract unit of user-service interaction that is stable over different types of user interfaces as well as different types of services (Nylander and Bylund 2002). No presentation information or otherwise modality or device specific information is included in the interaction act. There are eight different categories of interaction acts supported in UBI: `input`, `output`, `selection`, `modification`, `create`, `destroy`, `start`, and `stop`, where `input` is input to the system, `output` is output to users, `selection` is selection from a set of alternatives, `modification` is modification of entered information, `create` is creating new objects to be saved by the application, `destroy` is deleting objects, and `start` and `stop` starts and stops the interaction with the service. Interaction acts can be named and grouped, and groups can be nested. The interaction that a service offers users is described using interaction acts, thus keeping the description free from device and modality information. Performed user interactions are also encoded as interaction acts before they are returned to the service.

2.2 Customization Forms

Customization forms are optional in the Ubiquitous Interactor, and contains service and device specific information of how user interfaces should be rendered. The form can contain mappings to templates, media resources, color specifications, or other presentation information. Mappings can be based either on the name or the type of the interaction act, or both in combination. This way, different presentations of the same interaction act in a user interface can be created. By providing a customization form, service providers can control the presentation of a user interface in detail, for example giving it a certain look-and-feel. If no customization form is provided, a user interface is generated with default settings.

3 System Implementation

The Ubiquitous Interactor is composed of two parts: an *interaction engine*, and an optional *customization form* (see figure 1). Interaction engines interpret user-service interaction encoded in interaction acts, combine them with customization form information if present, and generate a user interface. Customization forms contain presentation information. Modules for generating and parsing interaction acts, and communicating between services and devices have been developed.

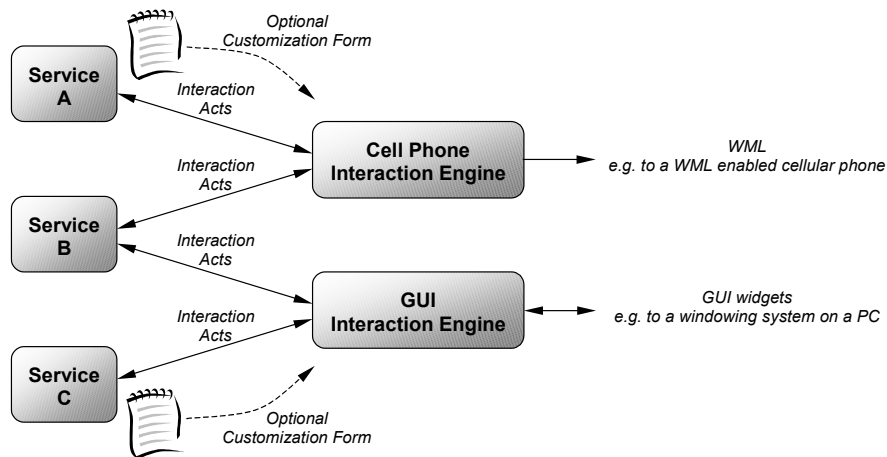


Figure 1: A design overview of the Ubiquitous Interactor system.

3.1 Interaction Engine

Interaction engines are specific both to device and to type of user interface, for example a GUI on a PDA. The task of interaction engines is to interpret interaction acts from services, combine them with device and presentation information from a customization form if there is one, and generate user interfaces of a given type for a given device. Since interaction engines are created for one type of user interface on a given device, or family of devices, the basic adaptation to device capabilities is built in. An interaction engine for a device without speaker and microphone would not generate user interfaces with sound based operations. Interaction engines contain default renderings for each interaction act, which are used when no customization form is provided.

Devices that can handle different types of user interfaces can have several interaction engines. We have developed interaction engines for Java Swing, std I/O, HTML, and Tcl/Tk user interfaces.

3.2 Customization Forms

The customization forms are implemented as a structured set of mappings between templates of user interface components on the one hand, and interaction act types and element names on the other. The structure can also contain media resources. Customization forms allows the generation of different user interface components for the same type of interaction act, based on the symbolic name of the interaction act. In our implementation, customization forms can be arranged in hierarchies, allowing one form to inherit mappings, resources, and links from another form. This allows for easy implementation of look and feel that is shared between several services.

3.3 Services

We have built a calendar service as a proof of concept. It provides simple standard calendar functions like adding and deleting meetings, displaying and editing added meetings, and show day, week, and month view of the information. The functions are accessible from three different user interfaces, a Java Swing UI and a web UI for desktop computers and laptops, and a Tcl/Tk GUI for PDA. All user-service interaction is encoded in interaction acts, and different

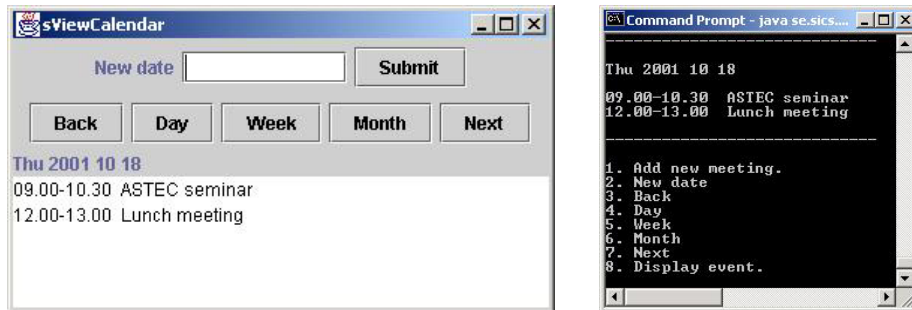


Figure 2: A Java Swing and a std I/O day view of the calendar

customization forms are provided for the user interfaces. For the Java Swing user interface, two different customization forms have been implemented. An example of different renderings could be a `selection` interaction act rendered as a panel with five buttons (back, day view, week view, month view, next) in one of the Swing UIs, as a pull-down menu in the other, and with only four buttons in the PDA UI (a decision on customization form level not to present a month view on the PDA). These different renderings are created from the same interaction act, combined with different presentation information. A performed choice in each of the user interfaces is encoded in the same way, so that no difference between the user interfaces is detected from the service side.

4 Related Work

Many approaches to achieve device independent applications have been proposed during the years, but the reasons behind the efforts have been different. During the seventies and eighties, device independence was seen as a way of overcoming different computer standards. This is the origin of User Interface Management Systems (UIMs), whose purpose was to shield the developer from device specific details during the development and maintenance work. Many of the UIMs focused on facilitating the development work, but a few of them also addressed the device independence issue, for example UofA* (Singh and Greene 1989) and Mike (Olsen 1987). The UIMs got more or less outdated in the mid-eighties when the personal computer arrived. The hardware was standardized (screen, keyboard and mouse), and the graphical user interfaces worked similar in different operating systems.

In the mid-nineties a new approach to device independence arrived: the web (Berners-Lee, Caillau et al. 1992). Documents and home pages could be displayed on computers with different hardware and operative systems if they were encoded with a markup language (HTML), and the computer had a web browser installed. The browser handled the rendering of the information. This caused problems since many people wanted to control the presentation of their information, and used pixel detailed specifications, and plug-ins to achieve this (Esler, Hightower et al. 1999). New versions of HTML (W3C 2002), and the Cascading Style Sheets (Bos, Wium Lie et al. 1998) are trying to provide better means for controlling the presentation, but the problem remains.

By the end of the nineties, new types of devices emerged on the market: smaller laptops, PDAs, and cellular phones with web browsers. Once again, service providers faced many different devices with different capabilities. This time, the differences are mostly not due to lack of standards, but to design decisions based on different intended usage. The XWeb project (Olsen, Jefferies et al. 2000) has used a web browser approach to solve the problem, i.e. the client software is in charge of the rendering of the user interface; Hodes et al. (Hodes and Katz 1999) provides several pre-defined user interfaces to choose from; Unified User Interfaces (Stephanidis 2001) is an approach where the environment and the user actions are monitored to make the user

interface adapt and thus provide a suitable user interface. The World Wide Web Consortium also has a working group in device independence, which at the time of writing (february 2003) has issued a working draft identifying various challenges concerning both application and device side of device independent authoring (Lewis 2002).

5 Conclusions

We have presented the Ubiquitous Interactor (UBI), a system for universal access to mobile services. In UBI the user-service interaction is used as level of abstraction. The interaction is kept the same for all devices, and complemented with an optional customization form containing device and presentation information. This way, a service can present tailored user interfaces to different devices. We have also described a calendar service developed for UBI.

6 References

- Berners-Lee, T., R. Caillau, et al. (1992). "World-Wide Web: The Information Universe." *Electronic Networking: Research, Applications and Policy* 2(52-58).
- Bos, B., H. Wium Lie, et al. (1998). Cascading Style Sheets, level 2. CSS2 Specification, World Wide Web Consortium.
- Esler, M., J. Hightower, et al. (1999). Next Century Challenges: Data-Centric Networking for Invisible Computing. The Portolano Project at the University of Washington. The Fifth ACM International Conference on Mobile Computing and Networking, MobiCom 1999.
- Hodes, T. D. and R. Katz, H. (1999). "Composable ad hoc location-based services for heterogeneous mobile clients." *Wireless Networks* 5: 411-427.
- Lewis, R. (2002). Authoring Challenges for Device Independence, World Wide Web Consortium.
- Nylander, S. and M. Bylund (2002). Providing Device Independence to Mobile Service. 7th ERCIM Workshop User Interfaces for All.
- Olsen, D. J. (1987). "MIKE: The Menu Interaction Kontrol Environment." *ACM Transactions on Graphics* 5(4): 318-344.
- Olsen, D. J., S. Jefferies, et al. (2000). Cross-modal Interaction using XWeb. UIST 2000.
- Singh, G. and M. Greene (1989). A high-level user interface management system. CHI 89.
- Stephanidis, C. (2001). The Concept of Unified User Interfaces. User Interfaces for All - Concepts, Methods, and Tools. C. Stephanidis, Lawrence Erlbaum Associates: 371-388.
- W3C (2002). XHTML 1.0 The Extensible HyperText Markup Language (Second Edition), World Wide Web Consortium.