# Semi-automatic generation of device-adapted user interfaces

*Stina Nylander*
Swedish Institute of Computer Science
Box 1263
16429 Kista, SWEDEN
Tel: +46-70-353-0369
E-mail: stina.nylander@sics.se

**ABSTRACT**
I am exploring an approach to developing services with multiple user interfaces based on a high level description of the service. The description is made using interaction acts, which are primitives for describing user-service interaction in a device independent way. Device-adapted user interfaces are generated based on interaction acts in combination with device and service-specific presentation information. As a proof of concept, the approach is implemented in a working prototype that handles graphical user interfaces, web user interfaces, and speech user interfaces for our sample services. Future work will mainly focus on how users experience and make use of services with multiple user interfaces.

**KEYWORDS:** User interfaces, multiple user interfaces, mobile services, device independence

**INTRODUCTION**
Services need to be able to present themselves on different devices to face the growing number of computing devices that are available to users. In many cases, services only work with a specific device or a family of devices (e.g. PDAs), constraining users' choice and sometimes forcing them to use several different services for the same purpose (e.g., one calendar for the cell phone and another for the desktop) [9]. To avoid this, services need to be able to adapt their presentation to various modalities and devices.

I am working with an approach that separates the user-service interaction from the service presentation. The user-service interaction is described using *interaction acts* [5], that are units of description free from information about presentation, device, or modality. This way the service description can be used for many different devices without changes in the service logic, and services can be developed for an open set of devices. The general description can be complemented by service and device specific presentation information enclosed in *customization forms* [5]. Based on the interaction acts and customization forms, each device can generate a suitable user interface for a service.


UIST'05, October 23-27, 2005, Seattle, Washington, USA.
ACM 1-59593-023-X/05/0010...$5.00.

The approach has been implemented and proved feasible in the *Ubiquitous Interactor* system (UBI) [4, 5], which handles generation of user interfaces for different devices based on interaction acts and customization forms. User interface generators for Java Swing, Java Awt, Tcl/Tk, web user interfaces, and speech user interfaces have been implemented.

**RELATED WORK**
Much of the inspiration for the Ubiquitous Interactor (UBI) comes from early attempts to achieve device independence or in other ways simplify development work by working on a higher level than device details. Mike [6] and ITS [11] were among the first systems that made it possible to specify presentation information separately from the application, and thus change the presentation without changes in the application. However, they only handled graphical user interfaces.

In more recent times, the issue of developing services for many different devices has gained renewed attention with the emergence of mobile and ubiquitous computing, see for example [1, 2]. The PUC system [3] uses state variables to automatically generate user interfaces to home appliances such as stereos and VCRs. The XWeb system [7] uses a description based on data types that is interpreted differently by different clients. PUC and XWeb handle both graphical user interfaces and speech user interfaces, but none of them provides a mechanism to allow service providers to control the presentation of the user interface that compares to the customization forms of UBI. Other differences from UBI are that they use a lower level of abstraction (state variables and data types), and they target a narrow range of applications (home appliances and other control applications) with predefined user input. In UBI, we

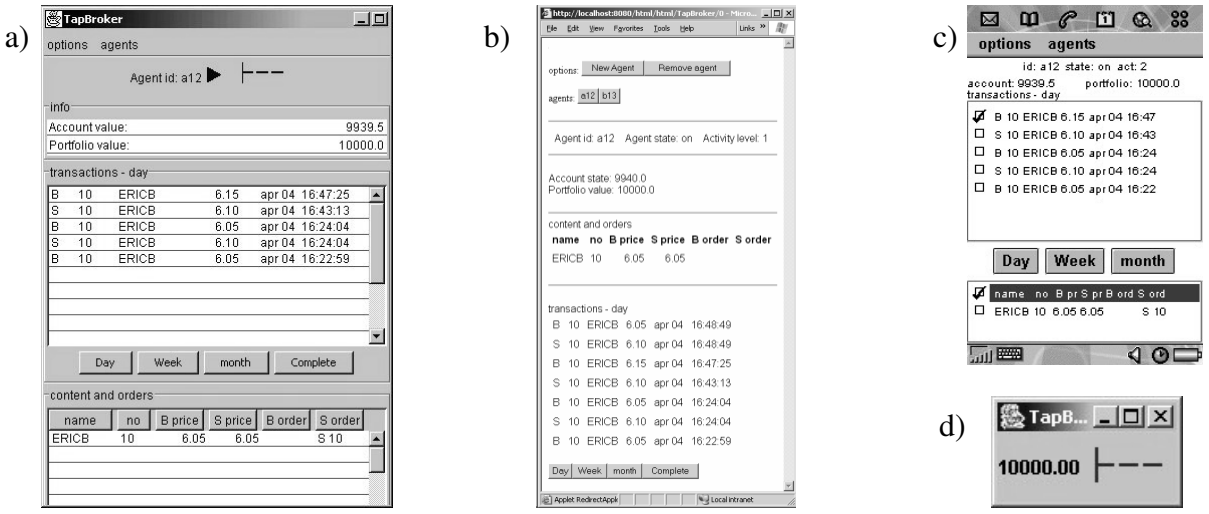


Figure 1: Example user interfaces to the stock brokering notification service generated with the Ubiquitous Interactor. All user interfaces are generated from the same service description. UI a) and d) are generated with a Swing interaction engine, UI b) is generated with a HTML interaction engine, and UI c) is generated with an AWT interaction engine.

have chosen to work with user-service interaction as level of abstraction to target a wider range of applications and to be able to handle free user input, for example notes.

## THE UBIQUITOUS INTERACTOR SYSTEM

The contribution of the Ubiquitous Interactor (UBI) is twofold: a conceptual part where the concepts of interaction acts, customization forms, and interaction engines are used for the development of services with multiple user interfaces; and a practical part where the concepts are implemented in a working prototype that serves as proof of concept. Two sample services have been created to show the functionality of the system; a calendar service and a stock brokering notification service.

### Interaction Acts

An interaction act is an abstract unit of user-service interaction that contains no presentation information at all. My approach builds on the assumption that user-service interaction for a wide range of services and devices can be captured with a small set of interaction acts in different combinations. UBI supports a set of eight interaction acts: *Start* and *stop* refer to the starting and stopping of services. *Create, destroy*, and *modify* refer to creation, deletion, and modification of service-specific objects, for example meetings in a calendar, or avatars in a game. *Output* is output to the user, *input* is input to the service, and *select* is selection from a set of alternatives. The last two are mainly used for data not stored in the service, such as data for navigation operations.

The current set of interaction acts is not intended to be exhaustive, but has proved sufficient for the information services that we have worked with. New types of services may require new interaction acts.

### Customization Forms

Presentation control is an important issue in commercial development [2], for example to brand applications. Customization forms are a means for service providers and service developers to specify how a service should be presented to end-users. By providing a detailed customization form, service providers have full control over how the user interface will be generated. Customization forms are optional. If no customization form is provided, or if the form is not exhaustive, defaults are used to generate the user interface. The main categories of presentation information in a customization form are *directives* and *resources*. Directives are mappings between interaction acts and widgets or other user interface components. Resources are links to pictures, sound, text, or other media resources that a particular user interface might need to present an interaction act.

### Interaction Engines

Interaction engines are service-independent but specific to a device or a family of devices, and to a type of user interface. For example, an interaction engine for HTML user interfaces could be used on both desktop and laptop computers, while handheld computers would need a special engine. Each device used for accessing a UBI service needs an interaction engine installed. In the ideal case, devices would be delivered with interaction engines pre-installed. Devices that can handle several types of user interfaces can have several interaction engines installed. For example, a desktop computer can have interaction engines for both Java Swing user interfaces and web user interfaces. During user-service interaction, interaction engines interpret interaction acts and customization forms (when available) and generate user interfaces for services. Interaction engines are also responsible for interpreting user actions and sending them back to services and for updating user interfaces. User interfaces can be updated both on initiative from services and as a result of user action.

### Implementation

The Ubiquitous Interactor is a working prototype with several interaction engines that handles the full set of

interaction acts. Interaction acts are encoded using the Interaction Specification Language (ISL) [5], which is XML compliant. Each interaction act has a unique id, a symbolic name, a life cycle value, a modality, an information holder, and a possibility to carry metadata. Customization forms are also encoded in XML. Each interaction engine contains modules for parsing ISL and customization forms, as well as generating responses to services from user actions. Interaction engines have been implemented for Java Swing, Java Awt, HTML, Tcl/Tk, and speech user interfaces. A calendar service and a stock brokering notification service [4] have been implemented as sample services. The calendar service has customization forms for a HTML user interface, a Tcl/Tk user interface (presented on a PDA), a speech user interface, and two different Java Swing user interfaces. The stock brokering notification service has customization forms for HTML, Java Swing (on a desktop computer), and Java Awt (on a cell phone).

## TAKING THE UBIQUITOUS INTERACTOR TO USERS

The main implementation phase of the Ubiquitous Interactor is completed, and its main purpose, to serve as proof of concept, is already achieved. The next phase is to evaluate the generated user interfaces. I will also use the knowledge that the implementation has given me about what we can do, and look at how users experience the concept of services with multiple user interfaces and benefit from it.

### User perception of UBI services– a pilot study

To find out how users were thinking about services with multiple user interfaces, we designed a two part pilot study. In the first part of the study, we used a variation on paper prototyping [8] where participants were instructed to create a GUI and a speech user interface for a calendar service using paper and pens. We explained to them that they were designing two user interfaces to the same service. In the second part they performed a set of tasks using a working GUI and speech user interface to a calendar service. We had eight participants working in pairs to make them communicate so that we could follow their thinking. The purpose of the first part of the pilot study was to encourage the participants to think and reason about services with multiple user interfaces, and the purpose of the second part was to give participants a sense of how a service like this could work in reality. We were not interested in their specific designs, the paper prototyping only served as a "thinking tool" for the participants. We interviewed them after each part of the study.

The paper prototyping really helped participants to think more generally about services with multiple user interfaces. They commented for example on information presentation that has to be much more concise in speech user interfaces since it is tedious and annoying to listen to long messages.

Most participants tried to make their two user interfaces as similar as possible. I think that for this purpose, making participants think and talk about services with multiple user interfaces, it might be a good thing to instruct them to make the user interfaces more different from each other. I also believe that the use of context information would help participants, both to understand the concept of services with multiple user interfaces and to paper prototype their user interfaces. In the pilot study, some participants had trouble understanding the reasons for having multiple user interfaces to services. Since their main computer experience came from desktop computers, they first thought that they were supposed to use the speech user interface with the desktop computer, which they found strange. Context information, maybe in the form of scenarios, could make it easier to understand that the different user interfaces are intended for use in different situations. The results from Truong et al. [10] also suggest that it is easier for users to talk about situations and tasks, than about devices and types of user interfaces.

## PROPOSED WORK

To complete my PhD, I will conduct a user study of services with multiple user interfaces. The main purpose of the study is to evaluate the user interfaces that I can generate with UBI. Feedback on how customization forms can be improved will be part of the result. A secondary purpose of the study is to investigate users' perceptions of, and experience with services with multiple user interfaces. A preliminary study plan is described below.

### Device adaptation vs. one UI fits (almost) all devices

The primary purpose of the study is to investigate which kind of adaptation of user interfaces that works best for services that are used from different devices. The two types of adaptation that will be investigated are the following.

The similarity principle – keep the user interfaces as similar as possible on all devices, even if it causes some trouble for the user, for example by making the user interface very small on a mobile device.

The device adapted principle – Adapt the user interface to the capabilities of the device (screen size, hard buttons etc.) but keep the structure and the capabilities of the service as similar as possible.

There are arguments for both principles. To keep the user interface similar, or even exactly the same, on all devices gives learning effects since users recognize themselves when they use a service on a new device. On the other hand, interacting with the service can be less smooth and practical, and some devices may be excluded. Adapting the user interface forces users to learn several user interfaces, but offers interaction that is well suited to the current access device.

### Study Setup

The study will be conducted in four steps. First a background survey of the participants will be made, checking their computer experience, their experience with mobile devices, and their experience with speech user interfaces. Second, participants will be asked to try out a service in the lab, using both device adapted user interfaces and similar user interfaces. Measures will be task

completion time, error rate, and subjective experience. Third, participants will be using a service "in the wild" for two weeks. They will be provided appropriate devices, and be instructed to use the service as they please during the trial period. After the two weeks, participants will be interviewed on their subjective experience of the trial. Forth, participants will be brought back to the lab to perform another set of tasks where task completion time and error rate will be measured, and they will be interviewed on their subjective experience.

The service for the study remains to be chosen, but possible alternatives are a calendar service or a game. It is important to choose a service that participants find meaningful to use during the "in the wild" trial, and both the calendar and a game could fulfill that purpose. The calendar since it is not difficult to find participants that need to do quite a lot of planning, and a game since well designed games are self motivating for interested participants.

## SUMMARY

The Ubiquitous Interactor is a system that provides concepts for developing services with multiple user interfaces, as well as a proof of concept prototype. The concepts are interaction acts that are used to describe the user-service interaction, customization forms that are used to provide service and device specific presentation information, and interaction engines that generate device specific user interfaces based on interaction acts and customization forms. The prototype implements the concepts and uses two sample services to show the functionality of the system.

Future work will concentrate on the evaluation of the device adapted user interfaces generated with the system, more specifically comparing them to services that present themselves with the same or similar user interfaces on different devices.

## ACKNOWLEDGMENTS

I would like to thank Markus Bylund, Annika Waern, and Magnus Boman for invaluable help and supervision during this work, as well as for co-authoring various publications on the system. I also want to thank Anna Sandin and Thomas Nyström for important help with the implementation.

## REFERENCES

1. Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J. and Zukowski, D., Challenges: An Application Model for Pervasive Computing. In Proceedings of 7th International Conference on Mobile Computing and Networking, (2000).

2. Myers, B.A., Hudson, S.E. and Pausch, R. Past, Present and Future of User Interface Software Tools. ACM Transactions on Computer-Human Interaction, 7 (1). 3-28.

3. Nichols, J., Myers, B.A., Higgins, M., Hughes, J., Harris, T.K., Rosenfeld, R. and Pignol, M., Generating Remote Control Interfaces for Complex Appliances. In Proceedings of 15th Annual ACM Symposium on User Interface Software and Technology, (Paris, France, 2002), 161-170.

4. Nylander, S., Bylund, M. and Boman, M. Mobile Access to Real-Time Information - The case of Autonomous Stock Brokering. Personal and Ubiquitous Computing, 8 (1). 42-46.

5. Nylander, S., Bylund, M. and Waern, A., The Ubiquitous Interactor - Device Independent Access to Mobile Services. In Proceedings of Computer Aided Design of User Interfaces, (Funchal, Portugal, 2004), 274-287.

6. Olsen, D.J. MIKE: The Menu Interaction Kontrol Environment. ACM Transactions on Graphics, 5 (4). 318-344.

7. Olsen, D.J., Jefferies, S., Nielsen, T., Moyes, W. and Fredrickson, P., Cross-modal Interaction using XWeb. In Proceedings of Symposium on User Interface Software and Technology, UIST 2000, (2000), 191-200.

8. Rettig, M. Prototyping for Tiny Fingers. Communications of the ACM, 37 (4). 21-27.

9. Shneiderman, B. Leonardo's Laptop. MIT Press, 2002.

10. Truong, K.N., Huang, E.M., Stevens, M.M. and Abowd, G., How Do Users Thing about Ubiquitous Computing. In Proceedings of Human Factors in Computing Systems (CHI), (2004), 1317-1320.

11. Wiecha, C., Bennett, W., Boies, S., Gould, J. and Greene, S. ITS: a Tool for Rapidly Developing Interactive Applications. ACM Transactions on Information Systems, 8 (3). 204-236.