SEMI-AUTOMATIC GENERATION OF DEVICE ADAPTED USER INTERFACES

Stina Nylander*

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 concentrate on evaluation of the generated user interfaces.

1. 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 the 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 one for the desktop) [10]. 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 [7], 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 a customization form [7]. Based on the interaction acts and customization forms, each device can generate a suitable user interface for a service.

The approach has been implemented and proved feasible in the Ubiquitous Interactor system (UBI) [6, 7] that 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.

2. 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 [8] and IST [11] were among the first systems that made it possible to

* Swedish Institute of Computer Science, Box 1263, 16429 Kista, Sweden

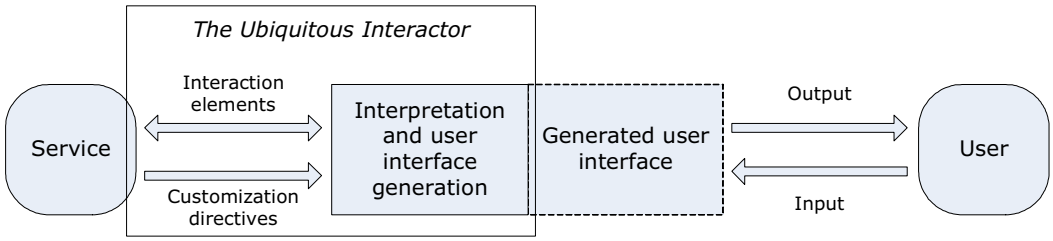

Figure 1: An architectural overview of the Ubiquitous Interactor.

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, 3]. The PUC system [4] uses state variables to automatically generate user interfaces to home appliances such as stereos and VCRs. The XWeb system [9] uses a description based on data types that is interpreted to different user interfaces 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 have a lower level of abstraction, and they target a narrow range of applications (home appliances and other control applications) with predefined user input. In UBI, we 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.

3. The Ubiquitous Interactor

The Ubiquitous Interactor (UBI) is a system that provides support for device independent development and use of services, *see figure 1*. In UBI user-service interaction is described in a general way, without any presentation information. Based on the general description, different user interfaces can be generated for different devices. When needed, device and service specific presentation can be fed into the process to tailor the user interface. Since the general description of the service stays the same for all user interfaces, service providers can develop services for an open set of devices. New devices or user interfaces can be added without changes in the service logic. It also makes it possible for service providers to control how their services are manifested for the end-users, which is very important.

The contribution of 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.

3.1 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 might generate new interaction acts.

3.2 Customization forms

Presentation control is an important issue in commercial development [3] 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.

3.3. 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.

3.4 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) [7], 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 [6] 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).

4. Proposed Work

The main implementation phase of the Ubiquitous Interactor is completed. I will continue to improve and maintain the system, but its main purpose is to serve as proof of concept and that is already achieved. Future work will focus on user evaluation, mainly regarding end-users, but also developers using the UBI concepts for creating new services.

4.1 Evaluation of end-user qualities

The end-user qualities that I set out to provide with UBI were device adapted user interfaces, and services with multiple user interfaces - i.e. the possibility to use the same service from different devices and situations. To evaluate to what extent these features appeal to users, and help them in their service use, user studies need to be conducted.

The device adapted user interfaces will be evaluated through a comparative study of on one hand user interfaces generated with UBI for different devices, and on the other hand both user interfaces created for a specific device and then moved to another device without changes, and user interfaces created "from scratch" for different devices. My hypothesis is that it is easier and more efficient to use services with user interfaces that are adapted to the device, i.e. the UBI user interfaces and the user interfaces created from scratch. The strength of the UBI user interfaces compared to user interfaces created from scratch will be that they are easier to create.

The scope and details of this study are yet to be defined, but potentially interesting parameters for adaptation are for example screen size, use of hard buttons and other external device features, and choice of modality. For screen size I would like to explore the effects of interacting with a user interface that does not fit the screen of the device, a small user interface on a desktop monitor or a large user interface on a small device. For the use of hard buttons I would like to investigate the effects of being able vs. not being able to use external buttons, scroll wheels and the like that most mobile devices are equipped with. For modalities I would like to explore the effects of being able to choose a suitable interaction modality, for example speech interaction in a mobile setting and a direct manipulation GUI in an office setting. Interesting differences to look for would be performance differences (do device adapted user interfaces cause less or more errors), differences in understanding (are services with device adapted user interfaces easier or more difficult to understand), and user preferences (do users prefer device adapted user interfaces or not). Obviously, the scope of the study needs to be narrowed down to a few of the aspects mentioned above.

The importance of being able to access the same service from different devices using device adapted user interfaces will be evaluated using qualitative methods. The system is not yet stable enough to allow real-life experiments so instead, I plan to use some more exploratory methods like focus groups to find services and situations where users believe that it would be useful to be able to access the same services from different devices. I am also interested in using role playing [2] to investigate how users perceive usage of services with multiple user interfaces.

4.2 Evaluation of developer qualities

The developer qualities will also be evaluated using qualitative methods. I expect the concepts of UBI to be helpful by supporting developers in considering services as device independent and therefore create service logic that is less tied to a user interface or an interaction modality. I also hope that the decoupling of the service logic will save development and maintenance work since there is only one version of the service logic to maintain. Given the results from a pilot study of developers working with UBI to create customization forms to existing services [5], it is not very informative to conduct laboratory studies where subjects work with UBI for a few hours. That way, only data about how novices understand the concepts is collected. A lab session is too short to allow developers to acquire any skill or habit in using the system and, thus, neither their performance nor their opinions are based on true knowledge of the system. Therefore, I will instead evaluate the aspects of UBI as a development tool by letting students work with the system for longer periods (typically six months for a Master's thesis project), follow their work closely, and then interview them. That way I will get valuable feedback on how useful the concepts are and what additional support is needed to help developers create services with multiple user interfaces.

The most difficult aspect to evaluate is to what extent the concepts of UBI save development and maintenance work. It is not possible to conduct studies where different groups of developers create and maintain services using UBI and traditional methods respectively and measure how much time they spend working with the services. It is more feasible to interview developers that have worked with UBI and ask them to compare UBI with other systems and development methods that they have used. However, all comparison with mature commercial development tools will be heavily affected by the fact that UBI is a research prototype, and that the concepts as well as the implementation might need to be refined. This evaluation will be the least prioritized among all the evaluation activities described here, not because it is the least important but because it is extremely difficult to obtain valid results.

5. 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 system, both in terms of end-user qualities and developer qualities.

6. Acknowledgements

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.

8. 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] Iacucci, G., Iacucci, C. and Kuutti, K., Imagining an experiencing in design, the role of performances. In Proceedings of NordiCHI, (Århus, Denmark, 2002), 167-176.

[3] 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.

[4] Nichols, J., Myers, B.A., Higgings, 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.

[5] Nylander, S. The Ubiquitous Interactor - Mobile Services with Multiple User Interfaces Department of Information Technology, Uppsala University, 2003.

[6] 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.

[7] 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.

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

[9] 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.

[10] Shneiderman, B. Leonardo's Laptop. MIT Press, 2002.

[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.