UPPSALA
UNIVERSITET

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 267

Design and Implementation of Multi-Device Services

STINA NYLANDER

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2007

ISSN 1651-6214
ISBN 978-91-554-6781-4
urn:nbn:se:uu:diva-7447

Dissertation presented at Uppsala University to be publicly examined in Auditorium Minus, Gustavianum, Akademigatan 3, Uppsala, Friday, February 9, 2007 at 10:15 for the degree of Doctor of Philosophy. The examination will be conducted in English.

Abstract
Nylander, S. 2007. Design and Implementation of Multi-Device Services. (Utveckling av tjänster med multipla användargränssnitt). Acta Universitatis Upsaliensis. *Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology* 267. *SICS Dissertation Series* 46. 46 pp. Uppsala. ISBN 978-91-554-6781-4.

We present a method for developing multi-device services which allows for the creation of services that are adapted to a wide range of devices.

Users have a wide selection of electronic services at their disposal such as shopping, banking, gaming, and messaging. They interact with these services using the computing devices they prefer or have access to, which can vary between situations. In some cases, the services that they want to use func-tions with the device they have access to, and sometimes it does not. Thus, in order for users to experience their full benefits, electronic services will need to become more flexible. They will need to be multi-device services, i.e. be accessible from different devices. We show that multi-device services are often used in different ways on different devices due to variations in device capabilities, purpose of use, context of use, and usability. This suggests that multi-device services not only need to be accessible from more than one device, they also need to be able to present functionality and user interfaces that suit various devices and situations of use.

The key problem addressed in this work is that there are too many device-service combinations for developing a service version for each device. In-stead, there is a need for new methods for developing multi-device services which allows the creation of services that are adapted to various devices and situations.

The challenge of designing and implementing multi-device services has been addressed in two ways in the present work: through the study of real-life use of multi-device services and through the creation of a development method for multi-device services. Studying use of multi-device services has gener-ated knowledge about how to design such services which give users the best worth. The work with development methods has resulted in a design model building on the separation of form and content, thus making it possible to create different presentations to the same content.

In concrete terms, the work has resulted in design guidelines for multi-device services and a system prototype based on the principles of separation between form and content, and presentation control.

Keywords: Human-Computer Interaction, Mobile Computing, Multi-Device Services

Stina Nylander, Department of Information Technology, Box 337, Uppsala University, SE-75105 Uppsala, Sweden

ISSN 1651-6214
ISSN 1101-1335
ISBN 978-91-554-6781-4
urn:nbn:se:uu:diva-7447 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-7447)

To all of you who took care of me when I could not take care of myself,
who believed in me when I did not believe in myself,
and who gave me a home when I did not have one…

without you I would not be where I am today.

Summary of included papers

Paper A: Stina Nylander. Real-Life Use of Multi-Device Services. SICS Technical Report T2006:18
This paper reports results from a user study of multi-device service use. Interviews were conducted with users of three different multi-device services to investigate practices, benefits, and problems with their use.

Paper B: Stina Nylander. Towards Design Guidelines for Designing Multi-Device Services. SICS Technical Report T2006:19
This paper presents guidelines for multi-device services based on analysis of a set of multi-device services.

Paper C: Stina Nylander, Markus Bylund, Annika Waern. The Ubiquitous Interactor - Device Independent Access to Mobile Services. In Proceedings of Computer-Aided Design of User Interfaces 2004. Eds : Rob J. K. Jacob, Quentin Limbourg, and Jean Vanderdonckt. Kluwer Academic media.
This paper describes the design and the implementation of the Ubiquitous Interactor. The main concepts of the system, interaction acts, customization forms and interaction engines are presented along with a detailed description of how they are implemented. Two sample services for UBI are also described, a calendar service and a stockbroker service. The design work presented in this paper was a joint work between the co-authors. Stina Nylander has done most of the implementation work and the writing of the paper.

Paper D: Stina Nylander, Markus Bylund, Annika Waern. Ubiquitous Service Access through Adapted User Interfaces on Multiple Devices. Journal of Personal and Ubiquitous Computing, 9(3), Springer Verlag.
This paper elaborates on paper C, giving more details on the design and implementation of the Ubiquitous Interactor. The design work presented in this paper is a joint work between the co-authors. Stina Nylander has done most of the implementation work and the writing of the paper.

Paper E: Stina Nylander, Markus Bylund, Magnus Boman. Mobile Access to Real-Time Information - The Case of Autonomous Stock Brokering. Journal of Personal and Ubiquitous Computing, 8(1), Springer Verlag.
This paper focuses on one aspect of the Ubiquitous Interactor: the ability to provide information push. The system features that make this possible are described, and the stockbroker service is presented in detail as an example of a service that depends on push of real-time information. Stina Nylander has done the design and implementation work with the TapBroker service, and most of the writing of this paper.

Paper F: Stina Nylander. Evaluating the Ubiquitous Interactor. SICS Technical Report T2003-19
In this paper the work with the Ubiquitous Interactor is evaluated, both in terms of the quality of the results and in terms of how well the research goals are achieved. A pilot study is presented and some pointers are given on how to proceed the evaluation.

Contents

Introduction

The goal of the work presented here is to find a method for development of multi-device services which makes it possible to create services that are adapted to a wide range of devices. To attain this goal, studies of how users handle multi-device services, identification of guidelines for the design of multi-device services, as well as technical work to create a framework for developing multi-device services have been carried out.

Users can choose from a wide selection of electronic services in areas such as shopping, banking, gaming, and messaging. They interact with these services using the computing devices they prefer or have access to, which can vary between situations. Sometimes a desired service does not function with the available device and users are forced to use a different service or to not use the service at all. Sometimes the desired service works with the available device, but the user interface does not suit the device and thus the worth of using the service is limited. To allow users to experience their full benefits, electronic services will need to be more flexible in the future. They will need to be multi-device services, i.e. be accessible from different devices. It has been shown that multi-device services are often used in different ways on different devices due to variations in device capabilities, purpose of use, context of use, and usability between the various devices (Paper A). This suggests that multi-device services not only need to be accessible from more than one device, they also need to be able to present functionality and user interfaces that suit various devices and usage situations (PaperD) (Trewin et al., 2003, Banavar et al., 2000, Shneiderman, 2002, Calvary et al., 2004).

The key problem addressed in this work is that there are too many device-service combinations to develop a service version that is adapted to each device. This would simply demand too much development and maintenance work. Creating a single version of a service that is accessible from all devices would also be difficult since devices have different capabilities. Furthermore, service use is different on different devices. Thus, there is a need to find methods for developing multi-device services that allow the creation of services that are adapted to various devices and situations without multiplying development and maintenance problems.

The problem of how to design and implement multi-device services has been addressed in two ways in the present work: through study of real-life use of multi-device services and through the creation of a development method for multi-device services. Studying use of multi-device services has generated knowledge about how to design multi-device services that provide users with good worth. The work with the development method has resulted in a separation between form and content that makes it possible to create different presentations of the same content. Automation is an important tool in this process but needs to be complemented with means of control (Calvary et al., 2004, Ponnekanti et al., 2001), given that automated user interface generation failed much due to the unpredictability of the generated user interfaces (Myers et al., 2000). It is important for service providers to be able to control the presentation of services (Myers et al., 2000, Esler et al., 1999) since the appearance of services is used e.g. for branding. Therefore, we have chosen to combine the automation with strong means for controlling the process of user interface generation.

The work has resulted in design guidelines for multi-device services (Paper B) and a system prototype based on the principles of separation between form and content, and presentation control. The system prototype, *the Ubiquitous Interactor* (Papers C-F) (UBI), provides a format for describing interaction between users and multi-device services based on a set of description units called *interaction acts*. They do not contain any presentation information and the user-service interaction can therefore be described in a device and modality independent way. The prototype has functionality for generating user interfaces from such descriptions. Device and service specific information can be fed to the process in the form of *customization forms* to gain detailed control of how the service is presented on a given device. This way it is possible to create different presentations for the same service without any changes in the service logic.

Definitions

The term *service* will in the present work follow the definition of Espinoza (Espinoza, 2003) where a service is considered a set of functions and abilities that manifests itself on a device. It is made available when needed, and might not reside locally on the device. A service distinguishes itself from an application by its loose coupling to the device. An application is installed on the device and executes locally while a service might be executing remotely as long as it is accessible from the device. With this definition of a service it comes naturally that the same service could be accessed from different devices and that the device is a portal to services (Banavar et al., 2000) (Saha and Mukherjee, 2003) rather than their “home”. A common example of a

service is a calendar that users can access both from their desktop computer and from their handheld device. However, we do not follow Epinoza in that services demand a particular payment model. In the present work, no assumptions on how, and if, users pay for their services are made.

A *multi-device service* is defined as a service that can be used from more than one type of device (desktop/laptop computer, PDA, cell phone), but only from one device at a time. Simultaneous use from more than one device is not required, neither is multimodal use.

The term *device* will denote an electronic apparatus that is used to manipulate something more than itself. Services or objects are manipulated through a device. Devices have user interfaces, and in some cases a device only supports one type of user interface while other devices (such as desktop computers) can support many different user interfaces.

Contributions

The contributions of this dissertation concern the process of both designing multi-device services and implementing them.

Two main contributions are made to the design process: empirical data on the real-life use of multi-device services, and design guidelines based on those empirical data as well as experiences from different research projects (Bylund, 2005, Nylander et al., 2005a, Johansson et al., Forthcoming, Nylander, 2006). The empirical data from study of several multi-device services show that service use is different on different devices and in different situations. Factors such as device capabilities, context of use, purpose of use, and usability are important and can support the design of multi-device services.

The contributions to the implementation process are both theoretical and practical. The main theoretical contribution is the identification of user-service interaction as a suitable level of abstraction between form and content, and a formal language to describe the interaction: the interaction acts. Interaction acts are units of description that do not contain any presentation information and therefore are independent of devices, services, and interaction modalities. This makes it possible to create many different user interfaces from the same set of interaction acts without any changes in the service logic. To give service providers the possibility to control the presentation of their services we have defined the concept of *customization forms* which contain service and device specific presentation information. By providing a

customization form, detailed control can be gained over the user interface that is generated from a given set of interaction acts.

The practical contributions to the implementation process are a system prototype and sample services that show that it is possible to generate different user interfaces to a service using interaction acts to describe the user-service interaction and customization forms to control the presentation.

Outline

The remainder of this dissertation can be seen as three sections. The first describes the theory and methods behind this work, followed by related research. The second section, Design of Multi-Device Services, describes the results from a study of multi-device service use and then presents design guidelines for multi-device services. The second section is based on paper A and paper B. The third section, Implementation of Multi-Device Services, describes the results from the technical work with the Ubiquitous Interactor. The system design and implementation is described, and a sample service is shown. The third section is based on papers C-F. The dissertation is concluded with a discussion of the work and some closing remarks.

Theoretical Foundation

The present work encompasses on the one hand the "traditional" human-computer interaction (HCI) which is mainly concerned with services designed for desktop computers and their usage, and on the other hand mobile computing which is a much more recent area in HCI. The traditional HCI has a well-established body of knowledge in design, usability, development, and evaluation (Preece et al., 1994), while the body of knowledge on mobile computing is still emerging, see for example (Weiss, 2002). Here, the two areas are combined into multi-device computing which includes the issues of traditional and mobile HCI, and the new issues of managing and combining services on multiple devices. This work also covers a smaller, but still important, part of the HCI research that is concerned with user interface development tools. Better tools and development methods are important factors in the HCI work for creating better services for end-users. If the time used for software development in a project can be reduced due to efficient development tools, then more time can be used for design and testing (Myers et al., 2000), something that could benefit end-users. The aim of the UBI work has been to create multi-device services that benefit users without creating extra development and maintenance work.

HCI has a strong foundation of working close to end-users during design, development, and evaluation. A certain phenomenon or problem is studied as close as possible to its natural settings to gather design information using a variety of methods such as participating and non-participating observation, surveys, and interviews. Some of the methods involve users actively, while others allow designers to gather information about the user situation without intruding. The HCI influence has also reached development work promoting methods like user-centered design (Norman and Draper, 1986) and participatory design (Schuler and Namioka, 1993) where users take part in iterative testing and modification during the development process. Intermediate and finished products are tested with end-users in laboratory experiments or field experiments.

We have not been able to adhere fully to the above described method, for a number of reasons. First, our research in the sView project (Bylund and Espinoza, 2000, Bylund, 2005) started out with a vision of how future electronic services could be used, thus there was no actual use or usage setting to study. Implementing the system showed that the vision was feasible and identified new research areas such as that of user interface generation. Second, the system has primarily been tested within our research group and it has been used in projects in other research groups (Bylund, 2005). It has not been tested with end-users since it relied on devices and infrastructure that were not available at the time. Third, a holistic view of the use of electronic services, such as the one presented with the sView system, needs a critical mass of services to show its power and potential in user tests. The goal of the project was to prove that the vision was feasible, thus developing a large number of services was outside the scope of the project. During the years that this work has been conducted, industry and the web community have come closer to our visions and at the time of writing it has been possible to for example conduct a study on real-life multi-device service use on publicly available services and end-users outside the research community (Paper A).

Methodology

This work covers several methodological areas. For the part that is concerned with the real-life use of multi-device services, methods such as interviews and heuristic evaluation of services have been borrowed from HCI. Prototyping and systems development have also been two important methods for the work presented in this dissertation, both when it comes to identifying the research problem and when it comes to solving the technical part of the same research problem.

The primary beneficiaries of this work are end-users and therefore it has been important to create a basis for the work on design of multi-device services in real-life usage. Methods and tools for developing multi-device services need to be grounded in usage rather than be technology driven. To get input to the technical development, it is important to find out how users are handling multi-device services. Multi-device computing takes place in various contexts which affects how services are used (Perry et al., 2001). Due to variations in context it is virtually impossible to even partially recreate the impact of context on service use in a research lab. Therefore, laboratory studies are not enough as empirical grounding for the design of multi-device services. Data from real-life use need to be collected even though it is difficult since it takes place whenever and wherever users find it suitable and not when researchers would like to study it. Moreover, there is always a risk that users get affected by the fact that they are studied and alter their behavior.

For the study presented in paper A, interviews were chosen as method for two main reasons. First, interviews make it possible both to collect information about issues relevant for the current study, and to identify questions for future work. Second, interviews give users the possibility to tell their story about motives, experiences, and problems using multi-device services. These two reasons were particularly relevant for this study since the study of multi-device service use is a new research area. It was important to identify the important research questions, and to get an understanding of how users perceive their situation.

A variety of methods have been used to study mobile service use which takes place at various locations and is carried out on small devices which makes it difficult to capture user actions camera. Methods that have been used are interviews, self reporting (Grinter and Eldridge, 2001, Palen and Salzman, 2002, Hulkko et al., 2004, Isomursu et al., 2004), non-participating observation (Weilenmann and Larsson, 2001), logging software (Demumieux and Losquin, 2005) and other techniques for capturing user actions. These methods are often combined (Grinter and Eldridge, 2001, Isomursu et al., 2004) since logging software and observation seldom give information about context or user intention, and self reporting and interviews often give unreliable information about how users actually interact with their services.

The technical work has its origin in the sView project (Bylund, 2005) that presented a vision of future use of electronic services. In the case of sView, prototyping and systems development were tools for explaining and demonstrating the vision of a new way of using electronic services as well as for showing technical feasibility and technical challenges.

The prototyping process for sView also revealed whole new research areas that needed to be investigated to fulfill the vision behind the system, one of them giving birth to the work with UBI. Again, prototyping became a tool for validating the approach and showing its feasibility. UBI has grown incrementally in many steps where the language for describing services and the functionality of the user interface generating system has expanded with every step. In the work with UBI, prototyping served both as a verification method and a dissemination method. By implementing new principles and ideas, we ensured that our approach had a solid foundation at all times. A working prototype proved the general feasibility and made it easy to demonstrate the concept of controllable user interface generation both to the research community and to industry partners.

Related Work

One of the contributions of this dissertation is the interaction acts, a set of units to describe the user-service interaction. Inspiration has come from earlier attempts to describe user-service interaction or user interfaces, some with the purpose of user interface generation and some with other purposes.

Foley et al. created sets of *interaction tasks* and *control tasks* (Foley et al., 1984) as a tool to help designers assign appropriate interaction devices to graphical user interfaces, e.g. mouse, light pen, and keyboard. Myers' *interactors* (Myers, 1990) were an effort to standardize user input to applications on a higher level. This way, developers would get device independent user input, and would not need to treat user input from various input devices. Neither Foley et al. nor Myers considered output in their categorization of interaction, and they were all limited to targeting GUIs. However, even though their goal was not user interface generation they provided the first steps in describing user-service interaction on a higher level.

Much of the technical inspiration for the Ubiquitous Interactor (UBI) comes from early attempts to overcome hardware diversities and achieve device independent applications or in other ways simplify development work. For a more comprehensive overview, see (Nylander, 2003).

MIKE (Olsen, 1987) and ITS (Wiecha et al., 1990) were among the first systems that separated form and content to make it possible to specify presentation information separately from the application, and thus change the presentation without changes in the application. Both were limited to graphical user interfaces, and imposed important restrictions on the interfaces they could generate. MIKE, for example, could not handle application specific data. In ITS, presentation information was considered to be application inde-

pendent and stored in style files that could be moved between applications. As pointed out by Wiecha et al. this was not an adequate approach. In UBI, we instead consider presentation as both application specific and tailored to different devices (Paper C, D).

With Selectors (Johnson, 1992), Johnson established a classification of interactive controls in the ACEKit based on application semantics rather than on appearance. The purpose was to go one step further than previous systems, and not only provide possibilities to change the look and feel of an application but also possibilities to specify the presentation for individual elements of the user interface. This corresponds to the way individual interaction acts can be mapped to different presentations in the Ubiquitous Interactor.

During the eighties, the hardware for the personal computer was standardized, and the need for device independent applications and methods to develop them diminished. The problem has returned with mobile and ubiquitous computing and the multitude of new computing devices. Again, service logic is separated from presentation to create device independent services. It is likely that standardization in hardware will appear in ubiquitous computing too, but the need for adapted multi-device services will persist. Usage will still be influenced by different contexts and different purposes of use which will pose different demands on services.

XWeb (Olsen et al., 2000) and SUPPLE (Gajos and Weld, 2004) encode the data sent between application and client in a device independent format using a small set of predefined data types, and leave the generation of user interfaces to the client. Unlike UBI, they do not provide any means for service providers to control the presentation of the user interfaces. It is completely up to the client how a service will be presented to end-users. In other words, these approaches enable device specific but not service specific presentations. Nichols et al. found when evaluating the PUC system (Nichols et al., 2002) that users expected an application to adhere to the look-and-feel of the current operating system, and added *smart templates* for that (Nichols et al., 2004). However, the adaptation allowed by the templates is quite superficial.

The Web has often been presented as a way of achieving device independent applications through separation between content and form. The content was structured with HTML and the formatting was left to the browser. Most devices can run a Web browser and thus access any service and generate a Web user interface. However, letting the browser generate the user interface provided poor control to the designer which led to more and more presentation information in the HTML code. The Cascading Style Sheets (Bos et al., 1998) were an effort to make it possible for designers to control the presenta-

tion of web pages without including presentation in the HTML code. The Web has some other drawbacks too. It can only provide page-based, user-driven interaction, which makes it less suitable for real-time applications (for example games). The device independence of Web pages can also be questioned. In many cases transformations or adaptations of pages are needed, for example to display a regular Web page on a handheld device with a smaller screen, and the research community has proposed many different solutions (Bickmore and Schilit, 1997, Lam and Baudisch, 2005, Baudisch et al., 2004, Trevor et al., 2001, Wobbrock et al., 2002, Menkhaus, 2002). To face the challenges of many different devices, the World Wide Web Consortium has created a working group addressing device independence for the Web. No recommendations have been issued yet, but two working drafts are published, one on content selection for device independence (DISelect) (Lewis and Merrick, 2005) and one on DIAL, the Device Independent Authoring Language (Smith, 2006). The combination of DISelect and DIAL allow service providers to specify alternative content to deliver to users based on their device. For example it would be possible to state that a picture will only be delivered to devices with a certain screen size while devices with smaller screens would get alternative content. This approach resembles UBI in that it makes it possible to create various service presentations to a single service. However, it differs from UBI in that their solution is to create a single service description that contains all the possible user interfaces. In UBI, we have chosen not to include the different variations in the interaction acts and instead provide presentation information and media resources separately in customization forms.

However, allowing separation of service logic and presentation is not enough for service providers. They also want to be able to control how their services are presented to the end-users (Myers et al., 2000, Esler et al., 1999). The user interface is the promoting channel for the provider, and it is important to be able to control that. Control of the presentation of user interfaces is provided in UBI, and also in the Unified User Interface system.

Unified User Interfaces (UUI) (Stephanidis, 2001) is a design and engineering framework for adaptive user interfaces. In UUI, user interfaces are described in a device independent way using categories defined by designers. Designers then map the description categories to different user interface elements. This means that designers have control of how the user interface will be presented to the end-user, but since different designers can use their own set of description categories the system cannot provide any default mappings. In UBI, we have chosen to work with a pre-defined set of description categories, along with the possibility for designers to create mappings. This makes it possible for the system to provide default mappings at the same time as designers can control the presentation of the user interface.

Other Related Research Areas

Other research areas that are not directly related to multi-device services but still are relevant here are adaptive user interfaces, context awareness, and model-based development. Below, they are presented and their relation to multi-device services described.

Adaptive User Interfaces

Adaptive user interfaces can change their behavior or appearance at run-time based on the user's interaction with the system either by maintaining a user model or by mechanisms for inferring patterns in user behavior (Schneider-Hufschmidt et al., 1993). Benyon (Benyon, 1993) argues for the use of adaptive systems in situations where variations in user behavior make it difficult to create a single design solution. An adaptive system can instead contain several designs for various tasks or users. Our approach to multi-device services also addresses variation, but differ from adaptive user interfaces in this sense on two main points. First, UBI user interfaces adapts to more than the user, mainly the device and its capabilities. Second, there is a technical difference. Most adaptive systems have predefined frames within which it can adapt based on user actions. UBI services have no predefined limits to what service presentations can be generated from a given set of interaction acts. This depends on the customization forms and generators that are created, and if new customization forms or new generators are created after the service is created, new unpredicted user interfaces can be created.

Context Awareness

Context aware services are generally defined as services that can gather information about their environment and react to changes in it without end-user input (Dey et al., 2001). The prime examples are location-based services that keep track of the users' location and adapt their content (for example by recommending nearby restaurants). The design of multi-device services as it is discussed in the present work does not include context awareness. Multi-device service design is primarily concerned with less dynamic factors such as device capabilities and the users' purpose for using a service. Context of use is also an important factor when designing multi-device services, but in more general aspects such as designing for mobility or limited attention, which change less often than users' location. These two areas can of course be combined, creating context aware multi-device services.

Model-Based Development

An early approach to creating services that adapt themselves to different devices was the model-based systems which strived for automatic user interface generation. Their goal was to allow developers to specify the user interface in a high level language and then automatically generate the user interface (Myers et al., 2000). Some of the systems had a separate device model and could thus generate different user interfaces for different devices (Wiecha et al., 1990, Olsen, 1987). The model-based approach did not catch on, however. Since the decisions of how to present the user interface were made by the system, based on for example a set of rules interpreting the high-level description, the appearance of the resulting user interfaces was unpredictable. The model-based systems could also only generate a limited range of user interfaces. Their failure adds support to our belief that although automation is a useful tool for generating user interfaces, the control of the result must lie with the service designers and service providers.

Designing Multi-Device Services

Multi-device service use is little studied, even though today it is possible to study publicly available services that users themselves have chosen to use on multiple devices every day. Real-life use of multi-device services provides valuable information on how to design multi-device services. In a study conducted by the author (Paper A) based on interviews with users of three multi-device services, study participants reported different usage patterns on different devices. They also reported other benefits with mobile service access than with desktop access, and different problems.

Findings from the study, analysis of a set of multi-device services, and experiences from several research projects (Bylund, 2005, Nylander et al., 2005a, Johansson et al., Forthcoming, Nylander, 2006)have been compiled into design guidelines for multi-device services. This chapter will first describe the differences in use between devices and then present the guidelines.

Real-Life Use of Multi-Device Services

The studied multi-device services were email, a teenagers' web community and a web dating service. Participants that used one of the services daily on both desktop computer and mobile devices were recruited.

Semi-structured individual interviews were made with 23 participants: seven email users, eight users of the teenagers' web community, and eight users of the dating service. The interviews were made in Swedish and all interviewed participants were residents of Sweden. About two thirds of the interviews were made over the phone due to large geographic distances. An interview form with open-ended questions was used, and participants were encouraged to elaborate their answers. Follow-up questions were asked when needed, as well as clarification questions. Each interview lasted about 30 minutes.

Multi-device service use has been little studied, and existing work has been conducted on research prototype services (Järvinen, 2005, Nikkanen, 2003, Marti and Schmandt, 2005). Studying a research prototype that was introduced to participants for the study implies that novice use will be studied, and also that participants may not have a personal motivation to use the ser-

vice. This will influence the results of studies of this kind. In several cases the focus also was on technical aspects of the systems rather than on usage (Nikkanen, 2003, Marti and Schmandt, 2005).

The remainder of this section will summarize the findings from the study.

Access and Awareness

Study participants reported the ability to easily check the state of their message box to be the most important benefit they gained when they could access their services from mobile devices (all three services provided messaging functionality). They carry their mobile devices all the time, and at any time they can check if any new messages have arrived. Since the mobile devices are always on and always connected through GPRS it is a quick and simple operation. Participants also stated that they often did not want to interact with their services from a mobile device in the same way as they do from the desktop computer, but to see if something had arrived or if they needed to do something. They wanted to check if they had received email or other messages, if they needed to answer (which they often chose to do from the desktop computer) or take some other action.

Some Contexts Favor Mobile Devices

The situations in which participants said they preferred to access their services from the mobile device could in many cases be characterized by lack of access to a desktop computer. However, sometimes discretion, simplicity, or comfort made participants choose a mobile device even though they had access to a desktop computer.

Participants reported that in some situations they chose to access their service from a mobile device even though they had access to a desktop computer. In some cases this was due to the discretion of the mobile device which makes it possible to access email during a meeting or during a family activity without being too obvious. In other cases it was considered easier to use the mobile device than going to the computer, for example while lying in bed, watching a movie, or cooking. The mobile device was also considered as more private and thus better to use when not wanting to share the service content with present friends.

Several of the participants had jobs that required a lot of movement, both within the city and over longer distances, for example visiting customers. When away from the office, they used their mobile device to access their email since they could not be sure when they would get to a desktop computer next time. In some situations, participants stated that it would be use-

less to bring a laptop computer since the situation would not allow placing it and using it anyway, for example when inspecting a construction site.

Participants also said that they were often engaged in other activities, for example at work, when using their services from a mobile device and thus had limited time and attention for the service. Another external factor that influences service use is cost.

Desktop Service Use Affects Mobile Service Use

Even though functionality on mobile devices was restricted for the services in the study compared to functionality on desktop computers, the desktop use "flows over" to the mobile devices. Functionality that is frequently used on desktop computers but not supported on mobile devices can cause problems in the mobile use in other ways than just being absent. A good example of this was email with attached documents. High end mobile devices can open MS Word and PDF documents but most of today's mobile devices cannot. Even the devices that can open documents usually work with low bandwidth so downloading attached documents becomes very slow. However, it is so common to attach documents to emails that it is almost impossible to offer mobile email without handling them in some way. This does not mean that all mobile devices should be able to open MS Word and PDF documents, but it is important to look at smooth ways to handle attached documents. For example allow for downloading the email but not the attached document to save time and money for the user. IMAP provides this, but also requires a server that supports it. The majority of the participants using email reported problems with attachments.

Different Activities on Different Devices

Several study participants stated clearly that they organized their use in such a way that some tasks were only attended to on the desktop computer, or that they strongly preferred to attend to them on the desktop computer and avoided them on the mobile device if they could. Tasks that require a certain overview, such as sorting emails into folders or looking at a personal presentation with text and pictures, were almost exclusively handled on the desktop computer. Tasks that were considered as central to the service, such as checking messages, were handled on both desktop and mobile devices. Browsing or other more unstructured use (surfing around) was done on the principal device which was the desktop computer for most participants but the mobile device for some of them. Many participants also reported that they tried to minimize input on the mobile device (see below in the section on usability).

Using different functionality on different devices as described above is connected to the capabilities of the devices and partly explains how a device gets the role of principal device. The majority of the participants considered the desktop computer as their main device for interacting with the case services. Choosing the desktop computer as the principal device was often motivated by its advantages in screen size and interaction possibilities. The principal device was used for unstructured use in situations that were not time critical (browsing around) and for more time consuming tasks.

Mobile Usability

Many participants reported usability problems with the mobile versions of the services. The main problems were related to the small size of the mobile devices and their limited input techniques.

The majority of the participants reported that they found input on mobile devices slow and tedious, and preferred to use the desktop computer with its standard keyboard for text input. If possible, they postponed writing messages or taking notes until they got to a desktop computer, and if they had to write on the mobile device they kept it very short.

Mobile devices have small screens and thus it is sometimes difficult to get an overview of service functionality or the service state compared to a desktop screen. Participants found it difficult to compose longer messages on mobile devices since they experienced difficulties seeing how the paragraphs would look on a larger screen. They also found that they got an immediate impression of the service and its functionality on the desktop screen since there is more space to present links and information, while a lot of navigation is required to obtain the same impression on a mobile device.

Guidelines for the design of Multi-Device services

As described above, device capabilities, usage situation, purpose of use, and service usability are factors that influence the use of multi-device services, and the differences in use between devices are notable. To design services for these conditions demands attention to specific problems such as how to adapt services to various devices, and service management on multiple devices (Paper A), as well as to the usual body of design knowledge. Based on those results and our experience from several research projects, we have formulated design guidelines for multi-device services (Paper B). These are high level guidelines focusing on the whole service, not just the user interface on the different devices. Guidelines on interaction design, graphic de-

sign, and usability can be found elsewhere; here we focus on the specifics of multi-device services.

Guideline 1: Create service versions that complement each other

Different devices have different capabilities and are thus more or less suitable to different tasks. When adapting a service to a new device it is important to take that into consideration and not try to squeeze in all service functionality. Taking advantage of the strengths of a device gives a better service than providing functionality that is cumbersome to use, even though only a subset of the functionality is available. Moreover, in the same way as it is important not to squeeze in service functionality in a device where it does not fit, it is not necessary to use every capability of a device. The home care service Joliv Mobile Omsorg described in (Johansson et al., Forthcoming) is an excellent example of a service that is better off without network connection even though it is used from network enabled devices.

Guideline 2: There should be overlap between service versions

It is important to keep an overlap in functionality between different versions of a multi-device service. Users of a service have knowledge about what their service can offer them and a new version of the same service should offer a subset of the functionality they are used to (and possibly new functionality too). If service versions do not have any functionality in common they risk being viewed as two different services by users, which could be confusing for users and negative from a service provider's point of view.

Guideline 3: Use context and purpose of use as design support

The context in which a new service version will be used gives valuable input to the design process. Some usage situations are connected to certain devices and certain service functionality and therefore can provide natural design choices, such as pointing out a subset of functionality to prioritize on a device. The purpose of use can provide the same support. The context of use can also provide more detailed guidance: for example users reporting that they do not have time to write notes in the mobile service (to difficult and slow with stylus) which indicates that only providing the standard input of the device is not enough. More support is needed, such as templates, speech-to-text or other means that make the input smoother.

Guideline 4: Do not forget usability

Multi-device services must live up to the same usability level as every other electronic service. For versions of multi-device services primarily targeting

desktop use there is a solid body of literature to guide the usability work. The usability work for mobile services is less mature but there are sources. Text input for example is known to be difficult on small devices and usability guidelines recommend keeping input to a minimum (Weiss, 2002). However, it is important not to remove the possibility. Most mobile email users prefer to write email on the computer and use the mobile device to check and read email, but they would not accept mobile email where it was not possible to write a reply when necessary.

Usability can be extra important in work situations where users have to use a service to perform their work. If such a service is difficult to use, the consequences can be severe for users.

Chapter Summary

The findings presented above strongly support the need for adapting multi-device services to various devices. Participants reported prioritizing different functionality on different devices. It has also been shown above that users do not have exactly the same purpose for each device when using a multi-device service, which also supports the need for adaptation. There are several reasons behind the differences in usage and purpose.

First, the context of use strongly affects service use. Participants of this study reported on for example the amount of available time or attention, accessibility, and cost as factors that influenced their choice of device and what functionality they chose to use in a given situation. In certain situations participants reported that they chose the mobile device even though they had access to a desktop computer. The most common reasons for choosing a mobile device in the presence of a desktop computer were that the mobile device was ready at hand, always connected, and more discreete.

Second, various devices provide different capabilities and advantages, which is an important factor when users decide how to interact with their services. Mobile devices are easy to keep at hand and can quickly provide state information about a service. Desktop computers have screen real estate that gives good overview and support more visual tasks, and offers easy input trough standard keyboards and mice. Device capabilities also control what functionality it is technically possible to provide on a device. Participants often stated that they preferred the desktop computer for browsing and organization tasks, while the mobile device provided quick access.

This suggests that design of multi-device services should not aim for the same functionality on all devices since the needs and uses are different for

the various devices. Instead it is important to take advantage of the strengths of each device. Mobile devices cannot compete with desktop computers in displaying data or providing overview, but they offer for example small form factor and means for notification.

Third, the usability of the mobile versions of the studied services was another factor influencing the usage. Many participants reported that usability issues made them avoid some tasks on the mobile device. Most notably, text input caused problems. The struggle for improving usability on mobile devices needs to continue.

These guidelines have been formulated to help designers keep these findings in mind when designing multi-device services, or, for example, when adding a mobile version to a service that has been designed for desktop use. However, it is important to note that following guidelines does not guarantee a successful result. Guidelines are only a complement, not a replacement of the understanding of users, technology, and their interaction.

Implementation of Multi-Device Services

To make it possible to create multi-device services that are adapted to many different devices, we need tools and methods to support their development. Creating a specific version for each device-service combination is not a viable option. Using the same version for all devices is not a good alternative since the differences between devices and situations of use are too large.

We have developed a method based on the separation of form and content that uses the user-service interaction as level of abstraction. By separating form and content it is possible to create many different presentations of the same content. We have also developed a system prototype, the Ubiquitous Interactor (UBI), that generates user interfaces based on an abstract description of the user-service interaction. The approach is semi-automatic generation, meaning that it is possible to add presentation information to the generation process to control the result. Sample services show that it is possible to generate user interfaces of different types and with different structure from the same abstract description.

The remainder of this chapter will further elaborate on our use of separation of form and content and the implementation of UBI.

Conceptual Separation

To create multi-device services we need methods and tools that support their development and allow for a certain degree of automation. We have chosen to work with a conceptual separation between form and content to create a technical framework for development of multi-device services. Separating form and content allows different presentations of the same content. Instead of having multiple versions of a service, the same service version can have multiple presentations. Development and maintenance work can be reduced.

However, separating form and content is not enough. Mechanisms for user interface generation are necessary as well as mechanisms to control the generation process. Having to make each change or each new user interface by hand is not an option even if form and content are separated. At the same time, fully automated user interface generation is equally undesirable, as

described above. We need partial automation that can help designers with adaptation and creation of multi-device services, and we also need room for design in the process. Therefore, we have chosen to work with semi-automation where user interfaces to services are generated from an abstract description of the service while it is still possible to feed presentation information into the process to control the appearance of the resulting user interface.

The purpose of separating form and content is to find a suitable intermediate level of abstraction where content can be represented without including elements of form. A representation of content that does not contain any information specific to a device or a type of user interface can serve as a base for adapting services to new devices and creating new user interfaces. The aim to separate form and content is certainly not new; it has been used to overcome diversity among devices and facilitate the development of applications for a long time. Early examples are Myers' interactors (Myers, 1990) and Foley's interaction tasks and control tasks (Foley et al., 1984) which tried to separate the form of input techniques from the application so that developers would only have to deal with input and not the different ways input was created. The Web is another well-known example which originally used HTML to mark up content and let browsers format the content and present it to end-users. As the Web and its content evolved, service providers took more and more control over the presentation of their content. As the range of devices used to access the web increased from desktop computers to handheld computers and cell phones, the smallest common denominator for all web browsers decreased, and many service providers now use a database approach to keep track of what markup to send to different user agents (browsers). Many solutions for how to transform standard web content to fit small devices have been proposed (Bickmore and Schilit, 1997, Lam and Baudisch, 2005, Baudisch et al., 2004, Trevor et al., 2001, Wobbrock et al., 2002, Menkhaus, 2002).

The work with UBI has its origin in the sView (Bylund, 2005) project which delivered a system prototype allowing users to gather a wide range of services in a personal environment. In sView, all services could share information about the user and users could administrate their services from a single place. The environment could migrate between devices through the network with unchanged service state and thus offer a continuous and seamless user experience in many different situations. To prove that it was possible for the service environment to follow the user to various devices and that users could interact with their services from these devices we needed to create not only working services but also a number of different user interfaces for each service, one user interface for each device. This quickly became cumbersome and time consuming, and the idea of generating different user inter-

faces from a single description surfaced, giving birth to the UBI project. We decided early not to go for a fully automatic approach since history had proved that automatic user interface generation was not very attractive (Myers et al., 2000). Our approach became to combine a device-independent service description with device specific presentation information to achieve controllable semi-automatic user interface generation.

In the present work, we have chosen the interaction between users and services as our level of abstraction between form and content in order to obtain units of description that are independent of device type, service type, and user interface type. Interaction is defined as:

> actions that services present to users, as well as performed user actions, described in a modality independent way (Paper D, p. 125).

Some examples of interaction according to this definition would be: making a choice from a set of alternatives, presenting information to the user, or modify existing information. Pressing a button or uttering a command would not be examples of interaction, since they are modality specific actions. By describing user-service interaction this way, the interaction remains the same regardless of which device is used to access a service. It is also possible to create services for an open set of devices. Interaction offers a high level abstraction that allows for large freedom in how to instantiate the units of the abstract service description into actual user interface elements. In return, the high abstraction level reduces the possibilities for fully automated user interface generation. Since our goal is to provide possibilities to control the adaptation of services, flexibility and control are more important than the ability for automatic user interface generation.

The Ubiquitous Interactor

We have realized our ideas of semi-automatic generation of user interfaces for multi-device services in a system prototype called the Ubiquitous Interactor (UBI). UBI provides controlled semi-automatic generation of user interfaces based on a separation of form and content. User interfaces are generated from an abstract service description, and presentation information can be fed into the process to control the result.

System Design

The user-service interaction in UBI is expressed in interaction acts that are exchanged between services and devices. In some cases the service in question will actually be running on the device, in other cases it might be on a

server. Interaction acts are interpreted by the device and user interfaces are generated based on interaction acts and additional presentation information. Whether services are running locally or on a server does not affect the way they express themselves, or the way interaction acts are interpreted.

Interaction acts are abstract units of user-service interaction that contain no information about modality or presentation. This means that they are independent of devices, services, and interaction modality. User-service interaction for a wide range of services can be described by combining single interaction acts and groups of interaction acts.

The latest set of interaction acts (see Paper D for details) that are supported in UBI has eight members: `input`, `output`, `select`, `modify`, `create`, `destroy`, `start`, and `stop`. `input` and `output` are defined from the system's point of view. `select` operates on a predefined set of alternatives. `create`, `destroy`, and `modify` handle the life cycle of service specific data, while `start` and `stop` handle the interaction session. All interaction acts except `output` return user actions to services. `output` only presents information that users cannot act upon.

In more complex user-service interaction, there might be a need to group several interaction acts together, because of their related function, or the fact that they need to be presented together. An example could be the create mail, reply, reply to all, and forward mail functions of an e-mail application. The structure obtained by the grouping can be used as input when generating the user interfaces. These groups allow nesting.

To allow for association of presentation information with interaction acts and groups of interaction acts, both groups and individual interaction acts have symbolic names that are used in presentation mappings. Groups and individual interaction acts can also be arranged in named presentation sets to allow for association of media resources to many interaction acts at a time.

In UBI, presentation information is specified separately from user-service interaction in customization forms. This allows for changes and updates to the presentation information without changing the service. The main forms of presentation information are *directives* and *resources*. Directives link interaction acts to for example widgets or templates of user interface components. Resources are used for providing pictures, sounds, or other media that are used to present an interaction act in the user interface. Both directives and resources can be specified on three different levels: group level, type level or name level. Information on group level affects all interaction acts of a group, information at type level provides information for all interaction acts of the given type; and information on name level provides information

about all interaction acts with the given symbolic name. The levels can also be combined, for example creating specifications for interaction acts in a given group of a given type, or in a given group with a given name.

It is optional to provide presentation information in UBI. If no presentation information, or only partial information is provided, user interfaces are generated with default settings. However, by providing detailed information service providers can fully control how their services will be presented.

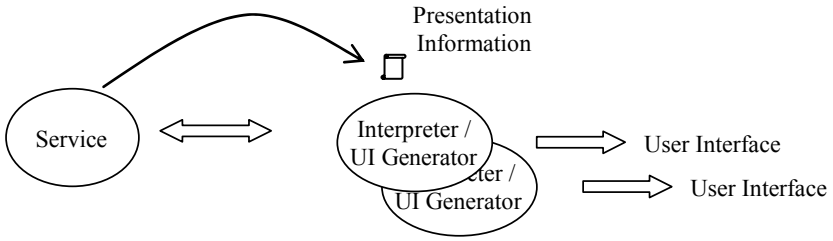

Figure 1. Services offer their interaction expressed in interaction acts, and an interpreter generates a user interface based on the interpretation. Different interpreters generate different user interfaces.

Based on interaction acts and customization forms, if provided, user interfaces to services are generated. Different types of user interfaces can be obtained by using different user interface generators, for example generators for Java Swing widgets, HTML, or speech would generate different user interfaces from the same set of interaction acts, see figure 1. Different user interfaces of the same type can be obtained by providing different customization forms.

System Implementation

The Ubiquitous Interactor has three main parts: the Interaction Specification Language, customization forms, and interaction engines. The Interaction Specification Language is used to encode the interaction acts sent between services and user interfaces, customization forms are used to control the presentation of user interfaces, and interaction engines generate user interfaces based on interaction acts and presentation information in customization forms. The different parts are defined at different levels of specificity, where interaction acts are device and service independent, interaction engines are device dependent, and customization forms are service and device dependent, see figure 2*Figure 2*.

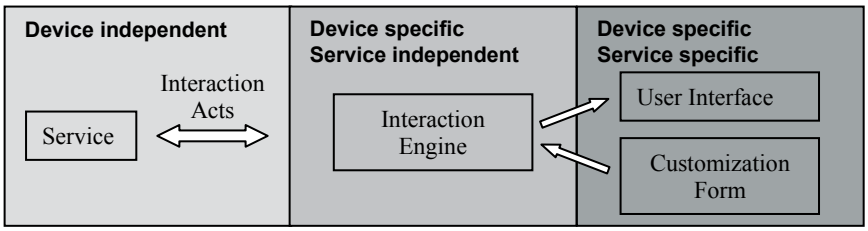

Figure 2. The three layers of specification in the Ubiquitous Interactor. Services and interaction acts are device independent, interaction engines are service independent and device or user interface specific, and customization forms and generated user interfaces are device and service specific.

The Interaction Specification Language is XML compliant and is used to encode information about interaction acts that is used in the user interface generation, such as id, name, group, life cycle, and modality. Customization forms are also encoded in XML, and use name, group, or type of the interaction acts to map presentation information to them.

We have implemented interaction engines for Java Swing, HTML, Java AWT, Tcl/Tk, and VoiceXML. These interaction engines can generate user interfaces for desktop computers. The default renderings of the Tcl/Tk interaction engine are designed to create user interfaces suitable for PDAs, and the Java AWT interaction engine has defaults for cellular phones of the Sony Ericsson P800/P900 type. The VoiceXML interaction engine generates speech based user interfaces (Nylander et al., 2005b). The system handles all interaction acts, and both directives and resources in the customization forms.

Service Example

One of the sample services that has been developed for UBI is a stockbroker service, the TAP Broker (paper F). It was developed as a part of a research project working with autonomous agents that trade stocks on the behalf of users (Lybäck and Boman, 2003). Each agent is trading according to a built-in strategy, for example buy low, sell high, or buy and hold (Boman et al., 2001), and users can have one or more agents trading for them. Since agents are autonomous, users cannot control them other than contacting the agent trade server manager and ask to have the agent shut down. Our service provides users with feedback on how their agents are performing so that they know when to switch agents, or shut one down.

The TAP Broker service provides agent owners with feedback on the agent's actions: order handling of the agent (placing and cancelling orders), and transactions performed by the agent (buying or selling stocks). It also provides information about the state of the agent: the account state (the amount of money it can invest), status (running or paused), activity level (number of transactions per hour), portfolio content, and the current value of the portfolio. However, it does not provide any means to configure or control the agent. The agents are created to work autonomously and cannot be manipulated from outside for security and fairness reasons.

We have implemented customization forms for Java Swing, Java AWT, and HTML (see figures 3 and 4 for sample pictures). For Java Swing, two quite different customization forms have been developed: one that generates a user interface appropriate for desktop screens, and one that generates a user interface for very small devices like Java enabled cellular phones. Since the screen size and presentation capabilities of desktop computers, PDAs and cellular phones are very different, user interfaces for the smaller devices only present parts of the available information.

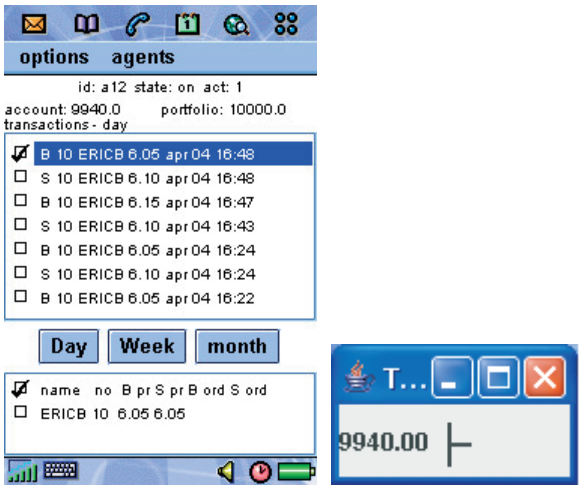

Figure 3. Two example user interfaces to the TapBroker service. To the left, a Java AWT user interface for a Sony Ericsson P800 smart phone. To the right a Java Swing user interface for a small device.

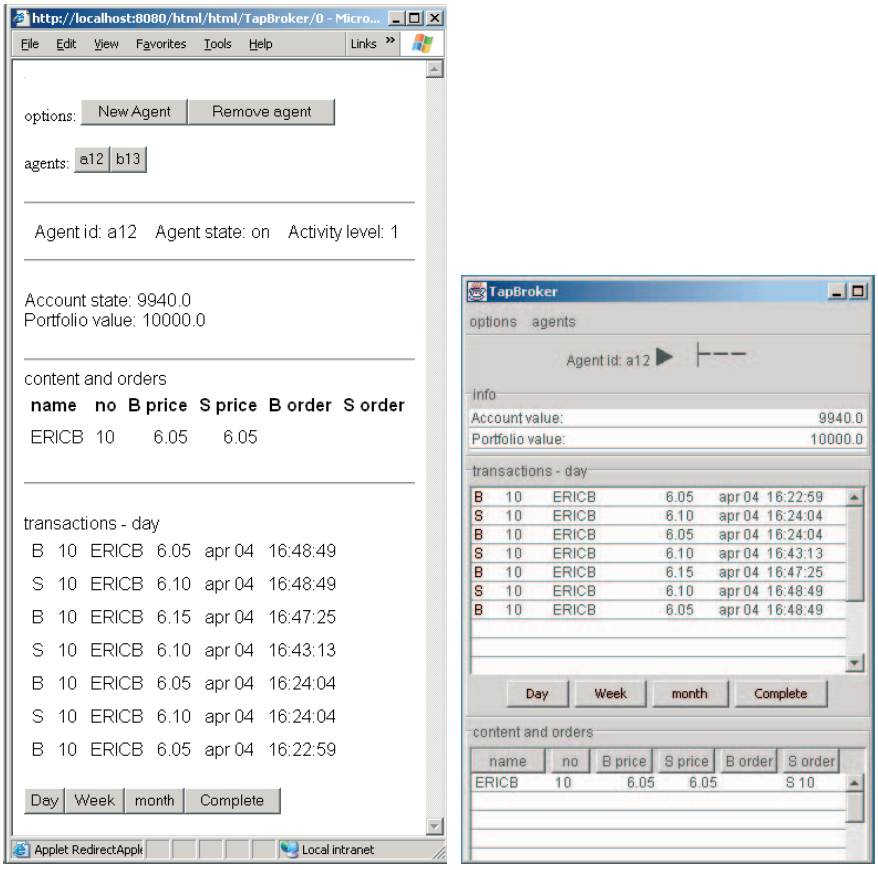

Figure 4. Two example user interfaces to the TapBroker service. The one to the left is an HTML user interface and the one to the right is a Java Swing user interface. Both user interfaces are designed for desktop computers. The user interfaces in figure *Figure 3* and*Figure 4* are generated from the same set of interaction acts.

Preliminary User Testing

We have conducted a small pilot study where we let four students work in pairs to develop Java Swing customization forms for the TapBroker (Paper F). The goal of the study was to find out if people that had not been involved in the development of UBI had problems understanding the concepts, and to collect information about how a larger study should be conducted.

The participants had no problems understanding the concepts of the system and the procedure of creating a customization form. However, neither of the two pairs got even close to creating a sufficiently complete customization form during the two hours. In the post-interview, the participants said that they got a good understanding of the basic principles of UBI. However, they

got a more vague understanding of the TapBroker service. They had no problems working with a service they had not developed themselves.

The experiences from this pilot study showed that participants needed more information about the service to fully understand it. For a larger study it will also be very important to recruit participants that are skilled in the user interface language and also to use a development environment that they are familiar with. Otherwise, programming problems risk hiding issues that are relevant for the study.

Closing Discussion

This work was initiated by our research group needing a method for creating several user interfaces to a single service. Throughout the work we have learned that when presenting a service on a new device, it is as important to present the right functionality on the device as it is to present it in the right way. It is not enough to transform one user interface into another. This is one of the reasons why we opted for a high abstraction level.

Context has also changed during this work. When it started in 2000, the range of handheld network capable devices was limited to a few WAP enabled cell phone models. Consequently, the number of existing multi-device services was small to say the least. Today, 2006, things are different. Not everyone owns a networked handheld device that lets them use multi-device services, and not everyone that owns one uses multi-device services, but quite a lot of people do both. This means that part of the sView vision is now reality; people can use at least some of their services from multiple devices. This development has made it possible to inform our work by studying actual use of multi-device services. It has also shifted the author's interests more towards user-oriented issues for future work.

This contextual change has introduced new parameters into the work. The starting point was mostly technical, and thus the focus was on device capabilities and differences between devices. This remains a strong issue but as the results of this work shows, studying the use of multi-device services has introduced new important issues such as context of use and purpose of use. Their importance for the design of multi-device services suggests that technical progress will not eliminate the need for adapting multi-device services. Differences in context of use and purpose of use will persist and thus the need for adapted multi-device services.

In the development of UBI, we have taken a clear standpoint in favor of providing developers with means for controlling the user interface generation process. We believe that automation is a powerful help in creating adapted multi-device services, but history has proved that automation without control is the wrong way to go. There are two main reasons behind the failure. First, lack of control. It is important for service providers to control how their product is presented to end-users. Second, the difficulties provid-

ing perfect defaults. It is very difficult to find presentation defaults for units of service functionality that result in adequate user interfaces for a majority of services, especially when dealing with service specific data. This means that there is no free lunch. We have to make a tradeoff between control and automation. We have chosen a solution where control has priority to be able to cover a wider selection of services. This means that we will seldom get the benefit of 100% automation, but in return we have a large freedom in the user interfaces we are able to generate.

Future Work

The existing set of interaction acts needs to be validated against other service domains. The sample services used in this work are information services, and other service domains could reveal the need for additional interaction acts. The subset of interaction acts handling service specific data, currently `create`, `modify`, and `destroy`, may also need to be refined to provide better basis for default presentations.

Further evaluation with developers is needed to assess this approach as a method for development of multi-device services. The aim is to reduce development and maintenance work compared to developing a separate version of a service for each device. To make such an evaluation possible more development work must be carried out. Libraries of presentations for the different interaction acts are needed to create realistic conditions for evaluation.

The real-life use of multi-device services provides important information to the design process and therefore needs further investigation. The study presented above discovered several interesting areas that merit more attention, such as how the context of use affects multi-device service use, what functionality should be assigned to what device, and how to solve the problem with overflow of use between devices. Other interesting issues that have caught the authors' attention during this work are how to convey community awareness on mobile devices and why the mobile version of a service sometimes becomes the primary version.

Summary and Concluding Remarks

The goal of this work has been to find a method for developing multi-device services. A study of users of multi-device services showed that their service use was different on different devices. Different functionality was prioritized on different devices and participants also reported on different purposes for using a service on different devices. Based on these findings, and experi-

ences from previous research projects, guidelines for multi-device services have been formulated.

A method for creating multi-device services has been developed. It is based on the separation between form and content to make it possible to create more than one presentation for a single service. User-service interaction has been defined as the level of abstraction, and is described using interaction acts, *i.e.* description units free from presentation and modality information. From a set of interaction acts, different user interfaces can be generated. To control the resulting user interface, it is possible to provide presentation information in the form of customization forms.

It has been shown here that UBI provides a method for developing multi-device services. It is possible to create multiple user interfaces from a single set of interaction acts and it is possible to control the resulting user interfaces through customization forms. This is an important step towards more flexible services. The need for adapted services will remain even if standardization will emerge in the mobile computing area just as it did in desktop computing during the eighties. The variations in context of use and purpose of use will remain and call for adapted services.

So, what will the future of multi-device services look like? I strongly believe that web user interfaces will be the dominating way of interacting with multi-device services. Web user interfaces have, wrongly, been considered device-independent for quite some time already. However, with the W3C working group for device independence important steps have been taken in the right direction. Moreover, given the recent evolution of the web, also called Web 2.0, web user interfaces have gained power and flexibility. However, web user interfaces compose a more limited approach to interacting with multi-device services than an approach like UBI could offer. For example the range of possible user interfaces would be limited to various markup languages.

Regardless of whether I am right or wrong about the future domination of the Web, multi-devices services will exist in the future and people will use them. Therefore, let us hope that future multi-device services will make us happy rather that driving us crazy…

Utveckling av tjänster med multipla användargränssnitt

Syftet med denna avhandling är att ta fram en metod för att utveckla elektroniska tjänster som gör det möjligt att skapa tjänster som är anpassade till många olika sorters datorer. Att utveckla en version av en tjänst för varje sorts dator blir ohållbart i längden eftersom antalet versioner då kan bli mycket stort.

En dator är inte längre inte bara den traditionella skrivbordsdatorn, utan tar sig många olika former. Vi har till exempel handdatorer och mobiltelefoner som fungerar som små, mobila datorer som delvis kan erbjuda samma saker som en traditionell skrivbordsdator. Termen apparat kommer i denna text att beteckna alla dessa typer av datorer.

I detta arbete definieras tjänst som en mängd funktioner som manifesterar sig på en apparat. Den finns tillgänglig när den behövs, men är inte nödvändigtvis lokalt installerad på apparaten. En tjänst skiljer sig i detta avseendet från en applikation som är installerad och kör lokalt på en viss apparat. Tjänsten är inte lika tätt knuten till apparaten.

Idag finns ett stort utbud av elektroniska tjänster. Vi kan handla en mängd olika sorters saker, uträtta bankärenden, spela spel och skicka meddelanden till vänner med hjälp av elektroniska tjänster. Vi använder dessa tjänster från den apparat vi har tillgång till vilket kan variera mellan olika situationer. Ibland fungerar inte den tjänst vi vill använda tillsammans med den apparat vi har tillgång till. Ibland fungerar den, men har ett användargränssnitt som inte är riktigt anpassat för en sådan apparat och då begränsas vårt utbyte av tjänsten. För att användare ska få fullt utbyte av elektroniska tjänster i framtiden kommer tjänsterna att behöva vara flexibla och gå att använda från många olika sorters apparater.

Vi har jobbat på två olika sätt för att lösa problemet med hur tjänster ska kunna utvecklas för olika sorters apparater. Vi har dels tittat på användning av tjänster från olika sorts apparater, dels arbetat tekniskt med att ta fram en utvecklingsmetod. Studiet av hur tjänster används har givit kunskap om hur tjänster ska designas för många olika sorters apparater, och arbetet med ut-

vecklingsmetoden har resulterat i en separation mellan form och innehåll för att kunna skapa olika presentationer för samma tjänst.

Användning av tjänster från olika sorters apparater har studerats genom intervjuer med användare av tre olika tjänster. I studien framkom att tjänster ofta används olika på olika sorters apparater, bland annat beroende på apparaternas olika förmåga, olika användningssyften och olika kontext. Till exempel uppgav studiedeltagarna att de ofta håller på med något annat samtidigt som de använder en tjänst från mobiltelefonen. De uppgav också att de föredrar att skriva meddelanden från skrivbordsdatorn medan de gärna kontrollerar om de fått nya meddelanden från mobiltelefonen. Riktlinjer för design av tjänster som ska användas från många olika sorters apparater har formulerats baserade på studieresultaten samt analyser av ytterligare tjänster och erfarenheter från tidigare forskningsprojekt.

Vi har valt att skilja form från innehåll för att kunna utveckla tjänster för många olika sorters apparater för att kunna skapa olika presentationer för samma tjänst utan att behöva ändra i koden för tjänstens funktion. Som abstraktionsnivå har vi valt att använda interaktionen mellan användare och tjänst. Interaktionen kodas med hjälp av *interaction acts*, som är beskrivningsenheter som inte innehåller någon information om dator, modalitet eller presentation. En systemprototyp har utvecklats, the Ubiquitous Interactor, som kan generera användargränssnitt baserat på en uppsättning interaction acts. För att kontrollera utseendet hos de användargränssnitt som genereras finns även möjlighet att tillföra presentationsinformation till genereringsprocessen. Presentationsinformation kodas i separata filer kallade *customization forms*. Olika användargränssnitt kan skapas utifrån samma uppsättning interaction acts genom att låta olika gränssnittsgeneratorer tolka uppsättningen. Generatorer för Java Swing widgets, Tcl/Tk, HTML, Java AWT widgets, och VoiceXML har tagits fram. Generatorerna för Java Swing widgets och HTML skapar användargränssnitt som i första hand lämpar sig för skrivbordsdatorer. Generatorerna för Java AWT och Tcl/Tk skapar användargränssnitt som i första hand lämpar sig för handdatorer. Generatorn för VoiceXML skapar talgränssnitt. Genom att tillföra olika presentationsinformation kan samma gränssnittsgenerator skapa olika användargränssnitt. Exempeltjänster med flera olika användargränssnitt för olika datorer har utvecklats för att testa systemet och visa dess potential.

Acknowledgements

First, I would like to thank my supervisors Bengt Sandblad and Annika Waern for guiding me through this work, each one with their own strategy to keep me on track and make me finish.

Thank you to VINNOVA for funding parts of this work, and to Playahead and Mötesplatsen for important help with the data collection to Paper A.

Anna Sandin, Thomas Nyström, and Ola Hamfors have given me important practical help: Anna with the implementation of the HTML interaction engine, Thomas with the implementation of the VoiceXML interaction engine, and Ola with crucial advice concerning the sView system.

Many people have given me valuable feedback in different stages of my writing, thereby substantially improving the quality of this dissertation: Magnus Boman, Markus Bylund, Mats Carlsson, Jussi Karlgren, Marie Sjölinder and Martin Svensson at SICS, John J. Barton at IBM Almaden Research Center, and David Benyon at Napier University. Thanks for your time and effort.

I would like to thank Erik Klintskog and Per Mildner for excellent support and friendship both in Uppsala, in Kista, and on the battlefield. You have been both faithful supporters and worthy opponents at the coffee table, and you have been good friends. Too bad Erik has left SICS.

Special thanks to Markus Bylund who has followed every step of my work. You have been my best colleague and most faithful supporter for almost seven years. Without your patience, encouragement, and ability to clarify things on a whiteboard this work would not be finished today. We have had great fun together, and I have learned a lot from you. And you have shown over and over again that you can stand the pressure!

Stort tack till min familj som har stått ut med mina studier i åratal utan att gnälla eller ifrågasätta. Men nu är det slut. Tro mig!

Thank you to all my friends that have supported me through ups and downs over the years. Thanks for being there, in good times as well as bad times: Eva, Madeleine, Karin, Nils, Rigmor, Ullis, and Ingrid. And Evy-Ann, for being the best possible aunt ever!

Åsa has been my very best friend for more than 20 years. Thank you for all the patience, encouragement, pushing, and honesty you have shown me over the years. And for all the fun we have had. I am also very grateful to you and Pelle for keeping your home open to me, feeding me meatballs when ever I am hungry, and for trusting me with taking care of your children every now and then. Finally, I would like to thank Lukas and Hannes for being two of my very best teachers in how to have fun, and for having a fabulous talent for saving even the worst day of dissertation writing. You certainly have your priorities in much better order than I do...

References

Banavar, G., Beck, J., Gluzberg, E., Munson, J., Sussman, J. and Zukowski, D. (2000) Challenges: An Application Model for Pervasive Computing, In proceedings of 7th International Conference on Mobile Computing and Networking (MobiCom), pp. 266-274.

Baudisch, P., Xie, X., Wang, C. and Ma, W.-Y. (2004) Collapse-to-zoom: viewing web pages on small screen devices by interactively removing irrelevant content, In proceedings of Symposium on User Interface Software Technology (UIST), Santa Fe, NM, pp. 91 - 94.

Benyon, D. (1993) Adaptive Systems: a solution to usability problems, *User Modeling and User-Adapted Interaction,* **3**(1)**,** pp. 65-87.

Bickmore, T. W. and Schilit, B. N. (1997) Digestor: Device-Independent Access to the World Wide Web, In proceedings of 6th International World Wide Web Conference.

Boman, M., Johansson, S. and Lybäck, D. (2001) Parrondo Strategies for Artificial Traders, In *Intelligent Agent Technology*, (Eds, Zhong, Liu, Ohsuga and Bradshaw), pp. 150-159.

Bos, B., Wium Lie, H., Lilley, C. and Jacobs, I. (1998) Cascading Style Sheets, level 2. CSS2 Specification, W3C Recommendations, World Wide Web Consortium.

Bylund, M. (2005) A Design Rationale for Pervasive Computing - User Experience, Contextual Change, and Technical Requirements, Doctoral Thesis, Department of Computer and Systems Science, Royal Institute of Technology, KTH.

Bylund, M. and Espinoza, F. (2000) sView - Personal Service Interaction, In proceedings of 5th International Conference on The Practical Applications of Intelligent Agents and Multi-Agent Technology.

Calvary, G., Coutaz, J., Dâassi, O., Balme, L. and Demeure, A. (2004) Towards a new generation of widgets for supporting software plasticity: the “comet”, In proceedings of Engineering Human Computer Interaction and Interactive Systems, EHCI-DSVIS, pp. 306-324.

Demumieux, R. and Losquin, P. (2005) Gather Customer’s Real Usage on Mobile Phones, In proceedings of International Conference on Human Computer Interaction with Mobile Devices and Services, pp. 267-270.

Dey, A., Abowd, G. and Salber, D. (2001) A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications, *Human Computer Interaction Journal,* **16**(2-4).

Esler, M., Hightower, J., Anderson, T. and Borriello, G. (1999) Next Century Challenges: Data-Centric Networking for Invisible Computing. The Portolano Project at the University of Washington, In proceedings of 5th International Conference on Mobile Computing and Networking (MobiCom), pp. 256-262.

Espinoza, F. (2003) Individual Service Provisioning, Doctoral Thesis, Department of Computer and Systems Science, Stockholm University/Royal Institute of Technology.

Foley, J. D., Wallace, V. L. and Chan, P. (1984) The Human Factors of Computer Graphics Interaction Techniques, *IEEE Computer Graphics and Applications,* **4**(6)**,** pp. 13-48.

Gajos, K. and Weld, D. S. (2004) SUPPLE: Automatically Generating User Interfaces, In proceedings of International Conference on Intelligent User Interfaces, pp. 93-100.

Grinter, R. E. and Eldridge, M. (2001) y do tngrs luv 2 txt msg?, In proceedings of European Conference on Computer-Supported Cooperative Work (ECSCW), pp. 219-238.

Hulkko, S., Mattelmäki, T., Virtanen, K. and Keinonen, T. (2004) Mobile probes, In proceedings of Nordic Conference on Human-Computer Interaction (NordiCHI), pp. 44-51.

Isomursu, M., Kuutti, K. and Väinämö, S. (2004) Experience Clip: Method for User Participation and Evaluation of Mobile Concepts, In proceedings of Participatory Design Conference, pp. 83-92.

Johansson, N., Nylander, S. and Sandblad, B. (Forthcoming) IT Supported Work in Home Health Care - a Case Study.

Johnson, J. (1992) Selectors: Going Beyond User-Interface Widgets, In proceedings of Conference on Human Factors in Computing Systems (CHI), pp. 273-279.

Järvinen, T. (2005) Hybridmedia as a tool to deliver personalised product-specific information about food, Research notes 2304, VTT, Espoo.

Lam, H. and Baudisch, P. (2005) Summary Thumbnails: Readable Overviews for Small Screen Web Browsers, In proceedings of Conference on Human Factors in Computing Systems (CHI), Portland, Oregon, pp. 681-690.

Lewis, R. and Merrick, R. (2005) Content Selection for Device Independence (DISelect) 1.0, W3C Working Draft May 2005, W3C.

Lybäck, D. and Boman, M. (2003) Agent trade servers in financial exchange systems, *ACM Transactions on Internet Technology,* **4**(3).

Marti, S. and Schmandt, C. (2005) Active Messenger: filtering and delivery in a heterogeneous network, *Human Computer Interaction,* **20**(1-2)**,** pp. 163-194.

Menkhaus, G. (2002) Adaptive User Interface Generation in a Mobile Computing Environment, Doctoral Thesis, University of Salzburg.

Myers, B. A. (1990) A New Model for Handling Input, *ACM Transactions on Information Systems,* **8**(3)**,** pp. 289-320.
Myers, B. A., Hudson, S. E. and Pausch, R. (2000) Past, Present and Future of User Interface Software Tools, *ACM Transactions on Computer-Human Interaction,* **7**(1)**,** pp. 3-28.
Nichols, J., Myers, B. A., Higgins, M., Hughes, J., Harris, T. K., Rosenfeld, R. and Pignol, M. (2002) Generating Remote Control Interfaces for Complex Appliances, In proceedings of Symposium on User Interface Software and Technology, Paris, France, pp. 161-170.
Nichols, J., Myers, B. A. and Litwack, K. (2004) Improving Automatic Interface Generation with Smart Templates, In proceedings of International Conference on Intelligent User Interfaces, pp. 286-288.
Nikkanen, M. (2003) One-Handed Use as a Design Driver: Enabling Efficient Multi-channel Delivery of Mobile Applications., In proceedings of Mobile and Ubiquitous Information Access: Mobile HCI 2003 International Workshop, Udine, Italy.
Norman, D. and Draper, S. (1986) *User Centered System Design: New Perspectives on Human-Computer Interaction,* Lawrence Erlbaum Associates.
Nylander, S. (2003) Different Approaches to Achieving Device Independence - an Overview, Technical Report T2003-16, Swedish Institute of Computer Science.
Nylander, S. (2006) Real-Life Use of Multi-Device Services, Technical Report T2006:18, Swedish Institute of Computer Science.
Nylander, S., Bylund, M. and Waern, A. (2005a) Ubiquitous Service Access through Adapted User Interfaces on Multiple Devices, *Personal and Ubiquitous Computing,* **9**(3).
Nylander, S., Nyström, T. and Pakucs, B. (2005b) Generating Speech User Interfaces from Interaction Acts, T2005:13, Swedish Institute of Computer Science.
Olsen, D. J. (1987) MIKE: The Menu Interaction Kontrol Environment, *ACM Transactions on Graphics,* **5**(4)**,** pp. 318-344.
Olsen, D. J., Jefferies, S., Nielsen, T., Moyes, W. and Fredrickson, P. (2000) Cross-modal Interaction using XWeb, In proceedings of Symposium on User Interface Software and Technology (UIST), pp. 191-200.
Palen, L. and Salzman, M. (2002) Voice-mail diary studies for naturalistic data capture under mobile conditions, In proceedings of CSCW 2002, New Orleans, Louisiana, pp. 87-95.
Perry, M., O'Hara, K., Sellen, A., Brown, B. and Harper, R. (2001) Dealing with Mobility: Understanding Access Anytime, Anywhere, *ACM Transactions on Computer-Human Interaction,* **8**(4)**,** pp. 323-347.
Ponnekanti, S. R., Lee, B., Fox, A., Hanrahan, P. and Winograd, T. (2001) ICrafter: A Service Framework for Ubiquitous Computing Environment, In proceedings of Ubicomp 2001, pp. 56-75.
Preece, J., Rogers, Y., Sharp, H., Benyon, D., Holland, S. and Carey, T. (1994) *Human-Computer Interaction,* Addison-Wesley.

Saha, D. and Mukherjee, A. (2003) Pervasive Computing: A Paradigm for the 21st Century, *Computer,* **36**(3)**,** pp. 25-31.
Schneider-Hufschmidt, M., Kühme, T. and Malinowski, U. (1993) *Adaptive User Interfaces,* North Holland.
Schuler, D. and Namioka, A. (Eds.) (1993) *Participatory Design: Principles and Practices,* Lawrence Erlbaum Associates.
Shneiderman, B. (2002) *Leonardo's Laptop,* MIT Press.
Smith, K. (2006) Device Independent Authoring Language, W3C Working Draft May 2006, W3C.
Trewin, S., Zimmerman, G. and Vanderheiden, G. C. (2003) Abstract User Interface Representations: How well do they Support Universal Access?, In proceedings of ACM Conference on Universal Usability, Vancouver, Canada, pp. 77-84.
Trevor, J., Hilbert, D. M., Schilit, B. N. and Khiau Koh, T. (2001) From Desktop to Phonetop: A UI for Web Interaction on Very Small Devices, In proceedings of 14th Annual ACM Sympoisum on User Interface Software and Technology, Orlando, FL, pp. 121-130.
Weilenmann, A. and Larsson, C. (2001) Local Use and Sharing of Mobile Phones, In *Wireless World: Social and Interactional Aspects of the Mobile Age*, (Eds, Brown, B., Green, N. and Harper, R.) Springer Verlag.
Weiss, S. (2002) *Handheld Usability,* John Wiley & Sons.
Wiecha, C., Bennett, W., Boies, S., Gould, J. and Greene, S. (1990) ITS: a Tool for Rapidly Developing Interactive Applications, *ACM Transactions on Information Systems,* **8**(3)**,** pp. 204-236.
Wobbrock, J., Forlizzi, J., Hudson, S. and Myers, B. A. (2002) WebThumb: Interaction Techniques for Small-Screen Browsers, In proceedings of Symposium on User Interface Software and Technology, Paris, France, pp. 205-208.

Acta Universitatis Upsaliensis

Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology 267

Editor: The Dean of the Faculty of Science and Technology

A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title "Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology".)

Distribution: publications.uu.se
urn:nbn:se:uu:diva-7447

ACTA
UNIVERSITATIS
UPSALIENSIS
UPPSALA
2007