Elaborating a Framework for Open Human Computer Interaction with Ambient Services

Andreas Lorenz, Markus Eisenhauer, Andreas Zimmermann
Fraunhofer Institute for Applied Information Technology
Schloss Birlinghoven
53754 St. Augustin, Germany
{andreas.lorenz, markus.eisenhauer, andreas.zimmermann}@fit.fraunhofer.de

ABSTRACT

In a world of ambient services, the technology disappears into the surroundings until only the user interface remains perceivable by users. Most preferably, the interface to interact with an ambient service is separated from the device hosting the services, e.g. the user interface is running on a mobile device and connected with the devices in the environment. In our work we will define a framework enabling different ambient services to work together with different input devices and vice versa. The goal is to enable the user to employ the device that fits best to the current situation, personal capabilities and gusto. In particular the use of devices the user is already familiar with should be transferred to additional or new services.

1. INTRODUCTION

In an ambient intelligence world, devices work together to support people in carrying out their daily activities in an easy, natural way using information and intelligence that is hidden in the network connecting these devices. The technology disappears into the surroundings until only the user interface remains perceivable by users. Key terms describing ambient services are

- Embedded hardware integrated in the environment (smart devices, sensors, interaction devices),

- Seamless mobile/fixed communication and computing infrastructure (interoperability, wired and wireless networks) and

- Personalized human-computer interfaces.

In opposition to the desktop paradigm, in which a single user consciously engages a single device for a specialized purpose, someone interacting with ambient services engages several computational devices and systems simultaneously. In this work we will describe technologies to enable any device to be an interaction device for interacting with ambient services. Because we address a wide range of computing devices (mobile devices like mobile phone, Pocket-PC, voice recording devices, gesture-based interaction device like Wii-Controller, and other) we use the term *input device* for the device the user has at hand.

In this paper we will elaborate a framework that abstracts from concrete input devices. The paper covers the whole process from identifying the needs and requirements, definition of the terms and components to the specification of the framework.

1.1 Scenario

Burkhard, a business man always interested in the newest technology, went to his favorite media store to get the newest home cinema equipment. He has to listen to a long introduction from the shop assistant, because there are so many options; at the end, Burkhard has the choice between another remote control or just a Compact-Disc with software for any Java-equipped mobile device. Because his wife is already tired of the zoo of remote controls on the coffee table in the living room, Burkhard takes the CD and installs a small piece of software at his pen-enabled PDA, which enables the device to be the remote control of the home-cinema equipment. For his wife, he installs the same software on her mobile phone - because her device does not support a pen, she uses another version that is using the joystick instead of the touch-sensitive screen. With the software, everybody is able to elect the best device and the favorite input style. The input is automatically translated to control commands and sent to the environmental device, which reacts accordingly. From now, it will be an important feature of any new electronic product to work together with Burkhard's smartphone as control device, too.

The scenario illustrates the vision of replacing hardware remote controls with software applications to be installed on any existing device. Other scenario would envision high integration of different input modalities with one and the same service, for example to have a haptic remote user interface for one user and in parallel a speech control for another user with visual acuity. Both envisions the selection of any modality by the user without the need to change the service.

2. PROBLEM DESCRIPTION

One limiting factor for market penetration of ambient services is the dependency between services and required interaction devices and methods. In private environments, e.g. at home, it combines purchasing a new product with the accompanied requirement to learn operating a new device and a new interaction method. In foreign environments, private devices are completely useless because they are not able to interact with the environment. The provision of special purpose devices by the vendor has proven to be one of the most limiting factor for acceptance of public ambient services.

What device is used is defined by the vendor of the service - regardless on the capabilities of the user, her devices already available and the task currently performed. For remote interaction with a service this pre-selection of the input device(s) becomes obsolete. *Example: A Movie-Player is running on a PC with TV-output. For starting the play-*

back, the player understands clicking on the "Play"-Button with the mouse, entering "Ctrl-P" with a keyboard or saying "Play" into the microphone.

The interaction could be improved if the user is in position of decision-maker, mainly because of four reasons:

1. The service usually does not care about the physical device the user employed. The source can be any device that is able to deliver the input to the service.

   Example: To invoke the "play"-method it is equal to the Movie-Player if the user employed the mouse, the keyboard or the microphone to express the input.

2. The service usually does not care about the input modality the user has chosen. The user can select any style that can be transformed one-to-one into the right format.

   Example: To invoke the "play"-method it is equal to the Movie-Player if the user has pressed "Ctrl-P", clicked on the "Play"-button or spoke the command.

3. The input devices usually work independent of the performing services, in particular they do not care about the behavior of the service.

   Example: Whenever the user pressed "Ctrl-P", clicked on the button, or spoke "Play" neither of the keyboard, the mouse or the microphone cares about whether the movie starts playing or not. Either knows nothing about movies.

4. **The user cares about the device, the modality and the behavior of the service** depending on

   - Situation

     Example: In front of the PC, the user might take the mouse; sitting on the couch, the user might employ a spoken command.

   - Task

     Example: For starting the playback, the user might employ the speech recognition - for browsing the file system searching a movie, the user might switch to the mouse.

   - Preferences

     Example: The user might find it strange to chat with a computer system and prefer to use mouse or keyboard if possible.

   - Capabilities

     Example: A physical impaired user might not be able to operate a mouse but capable of speech.

With our work we address service developers who want to enable users employ any device as input device, and interaction developers who want to enable input devices to work together with environmental services. We will empower distributing input events from any source to any service currently enabled without loosing it's meaning to the service. The remote access is illustrated in Figure 1.
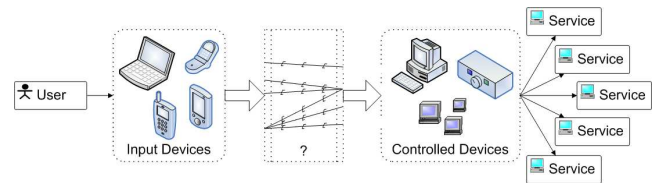

Figure 1: Transmission of user input from any remote device to service(s).

2.1 Requirements

The first step is to enable services to receive and consume input from devices a user already knows. The way to create desired input with the device, its meaning to the service, and the expected reaction from the service must be clear to the user. To be attractive to a wide range of users, the solution should work on different devices and in different environments. The communication of the device with the service must be wireless in order to be usable from any location.

The more developers are able to integrate their solutions, the more solutions will be available for selection. Usually, services from different vendors do not speak the same language. For integration we need to have a common way to express meaning of input events and a unique process for the exchange. The interface of the service should be consistent with available standards as much as possible. All specifications need to be well defined, first-class documented, easy to understand (here: "easy" from the perspective of a software-developer) and as low resource consuming as possible in order to be acceptable by developers. The implementation on both the input device and the service should be independent from operating systems and programming language as much as possible.

2.2 Goal

The goal of this work is to enable input devices to deliver user input to ambient services. The input device offers any kind of user interface meaningful to the selected ambient service(s). The input device transfers this information to the service which reacts in its specific way.

2.3 Research Statement

One-to-one semantic correspondence between remote input device and processing service is key to open Human-Computer Interaction with interactive ambient services with the user's choice of input devices.

3. RELATED WORK

System capabilities of ambient services are limited if the user is not equipped with a specific input device to have a channel for explicit input or controlling the behavior. To overcome limitations, the user could be equipped with a service-related device. The hand-out of devices from a vendor of the service boosts costs, requires the user to be willing to pick it up and requires trust into the user to bring it back at the end of the stay. The use of any private device is not possible today because they do usually not speak the language of the environment, not even in private settings.

The zoo of remote controls in home environments indicates high relevance of expressing input to remote devices. Excluding (wireless) mouse and keyboard input, the type
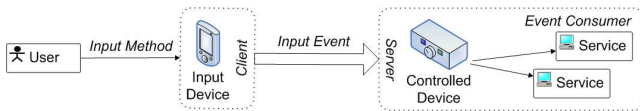
Figure 2: The Technical Components

of remote input can be categorized with rapidly decreasing percentage of use: Action events like On/Off, Up/Down, Play/Stop are present at almost all remote controls; Numbers and short texts are sometimes used, for example to operate the phone, switch TV-channels, or name movie recordings; hardly any use of longer text and pointer controls.

Iftode *et al* [5] identified the need for a simple, universal solution to control different applications in the environment of the user, which end-users are likely to accept easily. The input device should be programmable and support dynamic software extension for interaction with additional ambient services. For controlling the service, many approaches allow users to design their own remote control by creating new graphical interfaces that are downloaded to the input device after compilation. Beside these haptic input capabilities it is also possible to use speech recorded by a mobile device to control a remote system. Using for instance the Personal Universal Controller [8] a user can speak a command through which this is executed by the system. The focus is on automatic creation of the user interface from a description language.

Research in projects like IBM's "Universal Information Appliance" (UIA, [3]) or XWeb [9] results in the definition of a set of incompatible description languages like MoDAL (the XML-based language used by UIA) and UIML [1], where the programmer provides a specification (model) of the application, the display and the user. The concrete user interface is thus decoupled from the application, but only valid for a specific one. Though they are often based on similar components, they cannot be applied to another ambient service.

The iStuff Mobile architecture [2] is a platform combining (physical) sensor enhanced mobile phones and interactive spaces. The platform uses an Event-Heap [6] for distributing events of a specific type with specific fields. The mobile phone is then capable of sensing (local) user activity (e.g. key pressed) that are posted as (iStuff-)events on the heap.

3.1 Summary

Transferring events to any ambient service generally capable of performing the input but not able to correctly understand that particular type and content is useless. For development of GUI-based desktop applications there exist already common techniques and events that have a clear meaning to any application. For example, independent events like mouse-clicks can be delivered to any service; the mouse or the mouse-button itself does not know about the meaning to the application. The mouse could be replaced with any other physical device; if it fires correct **mouse**-events, the application will understand.

4. TERMS AND DEFINITIONS

For our work we identified the following components. Figure 2 illustrates their places in the overall process.

Input Device. The mobile device the user engages to control the ambient service. The type and shape of the device is not defined per se. On the remote device there is a client application running as the counterpart of the user-interaction. *Examples: Mobile phone, laptop, microphone, traditional computer systems.*

Controlled Device. A computing device that is available in the current environment of the user. *Examples: Small items providing information (sensors), output devices (displays, speakers), traditional computer systems.*

Ambient Service. An interactive application (short *service*) running on any controlled device. The user *consciously interacts* with the service in order to get information, adjust settings or control the behavior. *Examples: Small services providing information to the user (like the temperature), electronically actuate output devices (like playing an audio file at a speaker installation in a room), GUI-based applications on embedded devices (like showing a movie on the TV).*

Client. The role of the information provider in the client-/server approach. In our case, the client is the *application running on the input device*, receiving input from the user and delivering the input in a feasible manner to the event-consumer. *Examples: Networked application with graphical components, audio/voice-recognition systems, camera-based input, gesture recognition applications.*

Server. The role of an information receiver in the client-/server approach. In our case, the server is the *application running on the controlled device* providing any number of services. *Examples: Networked application with or without local user-interface, TCP/IP-Socket listener, Web-Server.*

Input Method. An abstract definition of *a way to express input by the user*, including the modality used with the input device. The potential meaning, interpretation and reaction depends on the service implementation. *Examples: Common GUI-based methods (for example clicking a certain button), voice commands (like spoken word "up"), gestures (like "Thumb up") or any other method. The examples could be interpreted to increase the volume by an audio player, but other services could implement their own interpretation (for example to move the cursor upwards).*

Input Event. The event delivered by the client to the server, containing the type of the event and other event-specific data. *Examples: Button-pressed events, key-typed events, or mouse-move events. The events include additional information like the character assigned with a pressed key or the position where a mouse click occurred.*

Event Consumer. A set of any number (including zero) of ambient services processing an event in their defined manner. If the set is empty, no service is able to make use of the information; the event is ignored.

5. DESIGN OF A FRAMEWORK

A software framework is an abstract design for a category of software systems. It defines a set of cooperating components, and the control flow in the system [7]. Some definitions implicitly require an object-oriented software design, defining a framework as an *"architecture of class hierarchies"* [10]. This might exclude efficient implementations based on other paradigms and get in the way of mixing up different software-designs for implementing specific components or operations. Because we are focusing on a *design* of solution(s), we will use the following definition of a framework:

Framework. A Framework is a generic architecture of a solution for a set of similar problems.

In computer science, the used term *architecture* is defined as the "*fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution*" [4].

In traditional (Graphical) User Interfaces (GUI), the user has access to application dependent graphic components. The user employs a mouse or keyboard in order to deliver input to the service. Most programming languages offer components to support the development of user interfaces, together with back-end-mechanisms for performing user input (for example event-listener mechanism). In this case, the service provides a performing method that is associated to input events occurring on the control component.

For the definition of our architecture, we abstract from the GUI-based control flow by defining a *Virtual Input Device*. It has the same attributes and behavior like any other device running a user interface except that it's shape is not defined; in particular, it does not necessarily provide any graphical representation. The only visible knowledge is a precise specification of input event(s) it delivers on the user's request. Potentially in parallel to existing graphical items, the developer registers listeners to the events for performing the input stream coming from virtual input devices.

The derived abstract system architecture is illustrated in Figure 3. The information flow to transfer an input event occurring on an input device to an ambient service consists of six steps to be performed: (1) Triggered by the user interaction, the client on the input device receives an input event from the local resources. (2) The input event is translated into a common representation. (3) The event is sent from the input device to the controlled device. (4) The input event is unpacked from its representation. (5) The input event is delivered to the target service(s). (6) The target service(s) consume the event. In this flow, the virtual input device covers the steps (2)-(5). For implementation we will use standard web-service technology defining input-adapters as web-services that are remotely executed over the network.

6. BENEFITS

The virtual input device covers the complexity of delivering input events from the client to the server. On the client side, the local event is handed over to a specific component as if the component was a local event consumer ("fire-and-
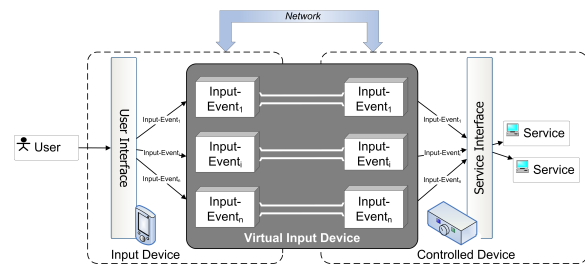

Figure 3: The abstract system architecture

forget"). On the server side, there is no difference between traditional event handlers and listeners to virtual input devices, which supports the integration between the two and transferring developer skills to the new setting. Because the *meaning* of an input event becomes independent from its representation, the creation of an input event is not bound to any static process: it is open to the developer, the capabilities of the device and the abilities of the user to employ any physical device and select the appropriate input method.

7. REFERENCES

[1] M. Abrams, C. Phanouriou, A. Batongbacal, S. Williams, and J. Shuster. UIML: An appliance-independent XML user interface language. In *International World Wide Web Conference*, Toronto, Canada, 1999.

[2] R. Ballagas, F. Memon, R. Reiners, and J. Borchers. iStuff Mobile: Rapidly prototyping new mobile phone interfaces for ubiquitous computing. In *SIGCHI Conference on Human Factors in Computing Systems*, pages 1107–1116, San Jose, CA, 2007. ACM Press.

[3] K. Eustice, T. Lehman, A. Morales, M. Munson, S. Edlund, and M. Guillen. A universal information appliance. *IBM Systems Journal*, 38(4):575–601, 1999.

[4] IEEE. Systems and software engineering - recommended practice for architectural description of software-intensive systems. *ISO/IEC 42010 IEEE Std 1471-2000*, pages c1–24, July 2007.

[5] L. Iftode, C. Borcea, N. Ravi, P. Kang, and P. Zhou. Smart phone: An embedded system for universal interactions. In *IEEE Intl. Workshop on Future Trends of Distributed Computing Systems*, 2004.

[6] B. Johanson and A. Fox. The event heap: A coordination infrastructure for interactive workspaces. In *IEEE Workshop on Mobile Computing Systems and Applications*, page 83. IEEE, 2002.

[7] J. Ludewig and H. Lichter. *Software Engineering*. dpunkt.verlag, Heidelberg, 2007.

[8] J. Nichols, B. A. Myers, M. Higgins, J. Hughes, T. K. Harris, R. Rosenfeld, and M. Pignol. Generating remote control interfaces for complex appliances. In *ACM symposium on User interface software and technology*, Paris, France, 2002. ACM Press.

[9] D. Olsen, S. Jefferies, T. Nielsen, W. Moyes, and P. Fredrickson. Cross-modal interaction using xweb. In *Annual ACM Symposium on User Interface Software and Technology*, pages 191–200. ACM Press, 2000.

[10] H. Züllighoven. *Object-Oriented Construction Handbook*. dpunkt.verlag, Heidelberg, 2005.