Extending Mobile Devices with Spatially Arranged Gateways to Pervasive Services

Dominique Guinard
Department of Informatics
University of Fribourg
Boulevard de Perolles 90
1700 Fribourg, Switzerland
dguinard@gmipsoft.com

Sara Streng
Institute for Computer Science
Ludwig-Maximilians University
Amalienstrasse 17
80333 Munich, Germany
sara.streng@gmail.com

Hans Gellersen
Computing Department
InfoLab21
Lancaster University
Lancaster LA1 4WA, UK
hwg@comp.lancs.ac.uk

Categories and Subject Descriptors

H.5.2 [**Interfaces and Presentation**]: User Interfaces - Interaction styles; C.5.3 [**Computer System Implementation**]: Microcomputers - Portable devices

Keywords

Spontaneous mobile interactions, nomadic user interfaces, pervasive services, sentient computing, pervasive services.

ABSTRACT

Discovery and interaction with services is as much a user interface problem as a networking problem. In this paper we introduce a system in which mobile devices are seamlessly extended with *Gateways* to services in their environment. The gateways are realized as widgets that are arranged around the edge of the mobile device's user interface to indicate the direction of discovered services, thus providing the user with an overview of interaction opportunities. The gateways also afford direct access to services integrate with the device's interface, for example supporting drag-and-drop of objects to a gateway and through the gateway to the corresponding service.

1. INTRODUCTION

Mobile users can potentially benefit from access to pervasive services that are physically manifest in their environment, such as display, printers, appliances and other computer- and network-enabled devices. Discovery and interaction with pervasive services has been primarily investigated from a networking perspective, concerned with how a user's device entering a network can become aware of the availability of useful services and how they can interoperate with identified services. We consider it of equal importance to view discovery and interaction with services from the perspective of the human user: how can a user become aware of the services that are available in their environment, and how can they invoke services and interact with them.

There are many ways in which mobile devices can facilitate or support their user's interaction with pervasive services. Generally these fall into two classes, the support for discovery of available services, and the support for access to and interaction with identified services. Examples for discovery support include use of Bluetooth, as available in many types of mobile device, to scan the environment for Bluetooth-enabled services. This approach is network-based and has the problem that there is no direct way for a user to associate names of discovered devices and services with actual devices in their environment. Other approaches to support interaction with pervasive services are based on some form of physical scanning, using mobile devices like a probe that is moved through an environment in order to find and use interaction opportunities. Examples are Fitzmaurice's pioneering work in which the mobile device is used like a lens [1], Rekimoto's Navicam which used augmented reality techniques [7], as well as more recent examples enabled by RFID- or NFC-technologies [8]. These approaches support physical identification of a service in the environment, but they do not help the user obtain an overview of services that are available in their proximity. We should also note other systems that use more bespoke techniques and metaphors for access to pervasive services through mobile devices, for example uPhoto which provides a "camera" to capture images that have embedded links to services in the photographed scene [3]. ICrafter takes a more programmatic approach and offers a framework to generate UI interfaces representing services in interactive workspaces [6].

As in ICrafter and Speakeasy [5] we introduce a system in which mobile devices are extended to support awareness of nearby services, as well as direct interaction with services through a novel kind of user interface. However, our system uses real-time relative spatial information: how far a service is located from the user's mobile device, and in which direction; this information can be obtained from an indoor location system, or by way of direct device-to-device sensing as demonstrated in related work [4]. This information is used to extend the mobile device's user interfaces with *Gateways* to services: these are small areas included at the edge of the screen, arranged to point in the direction of the discovered service, and serving as access point for interaction with the service. The spatial arrangement of the gateways around the user interface supports "visual service discovery" as opposed to "network service discovery", as it corresponds with the spatial arrangement of devices in the

Figure 1: The compass concept of the user interface.

Figure 2: User Interface after clicking on the printer gateway icon at the top of the screen. Note the interface also shows a gateway to a display available to the right.

environment as seen from the user's perspective. In earlier work we demonstrated a similar concept by providing mobile users with a 2D map view of co-located mobile devices as seen from their viewpoint [4]. The Gateway interface in contrast provide spatial references at the periphery of the user's interface, to be ready to hand but not in the way of other interactions with their device.

In this paper we report on the implementation of a proof-of-concept system including the seamless integration of Gateways in the mobile user interface, and the underlying infrastructure for access to services and interaction with services. The infrastructure allows data to be pushed "through the gateways" to discovered services to invoke default handling, and it also supports invocation of a service menu for other interactions a service may support. For our proof-of-concept, we demonstrate interactions with a public display, a printer, and a keyboard.

2. DESIGN OF THE GATEWAY SYSTEM

Our system brings two aspects together: the visual discovery of what is available (and where), and the spontaneous interaction with services that are available.

2.1 Visual Discovery of Services

In order to enable visual discovery of pervasive services we designed a user interface behaving like a compass. The interfaces shows gateways to services as small widgets arranged at the periphery of the screen. These gateways move as the user moves in the environment. Thus, if the user stands in front of a printer, a printer gateway will appear on the upper edge of the mobile display. Similarly, if the user stands left from a public display, the mobile user interface will show corresponding gateway on its left edge, facing toward the public display. As a concrete example of this concept consider Figure 1. The user is represented by the black spot in the middle of the first figure. In this situation, the printer is situated at the left of our user, the keyboard behind her and the display on her right. The application is aware of this context and will map the user's view of this environment on the user interface. As shown in the last part of Figure 1, three gateways are shown to the user on the mobile screen. The printer gateway appears on the left, the display gateway on the right and the keyboard gateway at the bottom.

Furthermore, two different modes of operation are provided for discovery. In the so called "scanning mode" the system displays a gateway for every device within the user's visibility range, to achieve a correspondence of what users see in front of them with what becomes identified through their interface. Alternatively, the "conditional mode" shows only services when certain spatial conditions are met. For instance, one could decide that the gateway to a public display is only shown to the user as long as they are facing it. Similarly, a gateway to a the keyboard might only appear when the user is close enough for direct interaction.

The gateways support two ways of interacting with the service they represent. First, the user can drag-and-drop objects onto the gateway area. For example, dropping a presentation on the public display gateway will start it on the display, and dropping a PDF file onto the printer gateway will invoke printing. Secondly, a service can be invoked using a click operation on the gateway area, in order to access a menu or dialog which will provide further service options. As shown in Figure 2, dropping a PDF file on the printer gateway will result in printing it with the default options, clicking on the gateway will open a print dialog on the mobile device to allow the selection of further options.

2.2 Spontaneous Interaction

The Gateway interface helps users make sense of devices and services that are available. In addition we provide an infrastructure that allows to interact with services, without any need for pre-configuration of interfaces. To facilitate this, a service is composed of a requester and a provider. The requester contains the information needed by a client to access the service, and the provider contains the actual logic. As shown on Figure 3 the service providers and requesters act as abstractions of the actual services. For example, when a user drags and drops a file onto the display gateway the UI checks whether it has a requester for the display. If this is the case the file is passed to the requester, which loads the file and sends it over the network to the provider. On the other end-point the provider running on a server gets the file and applies the core logic of the service using a concrete connector (or driver) to the final device.

Furthermore, each service can be either of type "push" or "pull-and-push". A push service describes a stateless, asynchronous service provided by a concrete device. A typical example would be a printer providing a service that prints a document using the default settings. The only data required from the client in such an interaction scheme is the file to be printed. Thus, the client can simply push the document to the other end-point and expect for it to be printed. The push services are quite interesting since they can be addressed in a uniform manner, using a single universal requester, without regards for the particular service semantics and core logic. Consider the printing service once again. If we now want to offer double-sided printing we need another type of service. Pull-and-push services offer to prompt the user for customized, statefull input before pushing it to the service provider.

This distinction at the service level permits the user interface to address all the requester in a uniform way, distinguishing them only by their method of invocation. Indeed, from the UI point of view push services are called using simple drag-and-drop, whereas pull-and-push services are invoked using a click on the gateway's iconic representation of the device.

3. PROTOTYPE IMPLEMENTATION

Figure 3: Invoking a service

Figure 4: Three concrete gateways.

3.1 User Interface

One of the most important requirements of the RelateGateways UI were integration and unobtrusion. In order to achieve these goals the interface is composed of small widgets integrated to the desktop. A view of the concrete implementation is shown as a detailed screenshot on Figure 2 or more concretely while running on a mobile device on Figure 7. Three types of GUI (Graphical User Interface) elements were designed:

- The visual representations of gateways are implemented as small windows in order to be able to move them at the screen periphery. As shown on Figure 4 each gateway contains an iconic representation of the actual device the gateway stands for. It helps the user in her visual discovery of the service and is used for the click invocation as well as for drag-and-drop interactions with the services.
- A toolbar offering various functionality such as starting and stopping the application, swapping between the conditional and the scanning mode, starting and stopping the service providers.
- A number of dialogs. These are used for user confirmation or user input (e.g. selecting the printing options) when a service is invoked.

Technically speaking, the user interface is implemented as a Java Swing GUI making use of the Swing Layout Extensions library. In terms of software components, a gateway is composed of:

- Two concrete views subclasses of VerticalView and HorizontalView. These are the visual representation of the gateways. The former is used when the gateway has to be mapped on the left or right side of the mobile display (see Figure 4, left and right), whereas the latter appears when the gateway has to be displayed on the top or bottom (Figure 4, middle).
- A GatewayController in charge of abstracting the positioning information in order to load and move the gateways' views at the periphery of the mobile screen.

Thus, the gateways' architecture implements the well-known MVC (Model View Controller) pattern. The VerticalView and HorizontalView of a gateway represent the views, the concrete GatewayController is the controller and the model is formed of the positioning information acquired from a sensing layer together with the service information.

3.2 Service Architecture

Our prototype system is implemented in Java SE. In our architecture, a concrete service extends the Service class and provide two components extending the ServiceProvider and ServiceRequester classes. Furthermore, each service has to be either of type PushService or PullAndPushService by extending the corresponding interface. As mentioned before, a PushService represents stateless and fully asynchronous service whereas a PullAndPushService is used to model statefull services requiring more user input than the object to apply the service on, also known as the service subject (e.g. the file to print, the presentation to start, etc.). This distinction is particularly interesting since it permits to create a "universal requester" for PushServices. Indeed, as such a service simply pushes the service subject, it does not need to know anything about the service logic and semantics. Thus, while each concrete PullAndPushService needs to offer a proper ServiceRequester to the mobile device, all the PushServices can be addressed using a single generic requester: the UniversalPushServiceRequester.

In order the understand the interactions of the software components, Figure 3 depicts the invocation of a presentation service. When the user drag-and-drops a presentation onto the display gateway the user interface has to send it to a service requester corresponding to the current device or to the universal requester. It will thus look for a concrete requester to invoke. Since all the requesters extend the ServiceRequester class the UI can address them in a uniform manner using the request(Object o) method they provide. The nature of the argument Object o depends on the type of service. The universal requester of a PushService (i.e. the UniversalPushServiceRequester) is called with a String object containing the local path to the dragged-and-dropped object. In turn, the particular requester of a PullAndPushService is called with a GatewayView object as parameter. Indeed, as futher user input is required when using a pull-and-push service, the requester needs to be able to address the gateway it was called from, for instance to pop up dialogs as on Figure 2.

Once invoked the UniversalPushServiceRequester connects to the PresentationProvider using a TCP/IP socket and sends it a copy of the drag-and-dropped presentation enclosed in a UniversalPushTransferObject. The provider is responsible for executing the service's logic. In order to reach this goal and make the services' logic as reusable as possible, a provider instantiates and uses a Connector. Such a software component is an abstraction of a service's logic. In this case an OpenOfficeConnector is instantiated. This latter class uses the Java UNO API[1] to connect to OpenOffice and eventually start the presentation on the public display.

In order to test the application, five concrete services are provided. They all use the architecture described above. Table 1 describes them into further details.

The concrete providers and requester form the networked

[1]http://api.openoffice.org/

Device	Name	Type of Service	Description	Concrete Connector
Printer	DirectPrinting	Push	Prints document with the printers' default options.	Java Print Service API
	CustomPrinting	Pull-and-Push	Prompts the user for file selection and open a printing option dialog box. Prints document with the selected options.	Java Print Service API
Display	PresentationShow	Push	Starts a presentation.	OpenOffice UNO API
	Presentation	Pull-and-Push	Pops up a file selection dialog and opens the presentation in OpenOffice Impress.	OpenOffice UNO API
Keyboard	Typing	Pull-and-Push	Redirects the typed keys to requesting device.	OpenOffice UNO API and Java AWT Event API

Table 1: Concrete services implemented for the tests.

Figure 5: User interface and Wizard of Oz interface.

Figure 6: Location of services in the test room.

communication end-points between the actors. Thus, an important assumption for our architecture is the fact that all the devices providing services as well as the mobile device have to be part of the same network, or at least have able to access one another over a TCP/IP network. Since nowadays a lot of mobile devices tend to have wireless connectivity we believe that this assumption is not excessively restrictive. Using relatively low-level sockets for service communication rather than higher-level networking schemes (such as RMI, CORBA, etc.) is a conscious choice. Indeed, we wanted the RelateGateway application to be as portable as possible in order to test it on various types of devices (handheld PCs, mobile phones, etc.) and allow the rapid prototyping of services. While using a Java implementation already helps towards portability, implementing the services on a socket basis reduces the language dependency, allows to think about cross-platform and cross-languages services and keeps the architecture simple and lightweight.

It is worth noting that no network discovery mechanisms are implemented in this prototype. As a consequence the IP addresses used by the requesters to connect to the providers are hard-coded in a configuration file used by the mobile devices. Furthermore, the requesters required to invoke pull-and-push services are bundled with the application on the mobile device. While this would certainly not be a good solution for a real application it is adapted to this early prototype. Indeed, the principal aim of this first version was to run a user study focusing on visual discovery and spontaneous interactions. Involving network discovery mechanisms at this early stage was neither necessary nor bringing a real added-value for the user study.

4. EVALUATION

4.1 Scenario

In order to evaluate the application we designed a small scenario. As shown on Figure 6, three devices are installed in a large room: a printer, a display, and a keyboard. Each device is attached to a computer. Each computer is then connected to a router coupled with a wireless access point in order to create both a wired and wireless local network. Each of the three devices provides various services as described on Table 1. Additionally, the Gateway system was installed on two types of mobiles devices: a OQO handheld PC (see Figure 7) and a Paceblade tablet PC both running Windows XP. It is worth noting that the application could as well run on a PDA or a mobile phone as long as these offer Java SE support.

Figure 7: RelateGateways application running on a Windows XP OQO.

4.2 Formative User Study

The study was conduct in two phase. The initial phase was essentially a test-run with five users intended to detect as many bugs as possible. Observations during this phase also permitted to improve some parts of the user interface. As an example, most user expressed a feeling of having to disconnect from the keyboard gateway once the service was consumed. This feature was not provided nor really required by the application but it was included in the next release. The interaction zones of the gateways were also redesigned after the first study. Indeed, the initial version comprised two distinct zones corresponding to the two interaction techniques exposed in Subsection 2.1: a drag-and-drop zone and a button. Since our five first users mentioned this fact as quite confusing we decided to redesign the gateways with a single zone (as shown on Figure 4) for both interaction schemes.

During the second phase, fifteen users evaluated the application in the room described above. The majority of recruited users were students with Computer Science background. Half of them were given an OQO handheld PC and the other half a tablet PC running the application. After a brief introduction to the project's context and field of use the user were asked to use the mobile device we provided them with in order to solve three tasks:

1. Discover a keyboard and use it in order to type some text in an instance of Open Office running on the mobile device.

2. Discover a public display and use it for starting a slide show stored on the mobile device.

3. Print a document located on the mobile device by discovering a printer and interacting with it.

As shown on Figure 5, a Wizard of Oz interface was used to track the user in the test room and to provide spatial information to our system, thus effectively serving as a very controlled sensing layer.

In general terms, the study revealed the users' excitement for these type of interactions: most of our test users commented this would be very useful in places they were not familiar with, which confirms our initial assumption. The fact that no installation or configuration was required to interact with the service was the most widely perceived benefit of the application. This shows, once more, the relevance of Weiser's vision [9] of the disappearing computer, in which the users' administrative tasks (e.g. device installation or configuration) are reduced to as few as possible. Our users expressed this fact as a requirement when it comes to visiting a new environment, which shows the pertinence of this assumption when addressing interactions with pervasive services.

The next most cited benefit was the dynamic spatial arrangement of the gateways. They commented it as making the mobile user interface more natural, "really smart" and quite helpfull for discovering the services in the room. This is not a surprise since it is known from various studies that humans structure their environment primarily spatially [2]. They also enjoyed being able to interact with all the services available using the same intuitive techniques. All users mentioned the drag-and-drop interactions across devices as being particularly appealing.

In terms of design issues a number of users discussed the size of the gateways. We designed them in order to be as unintrusive as possible on mobile devices, i.e. quite small. While this was fine with the users as long as they did not interact with the gateways, they suggested the system to somehow detect their will to use a particular gateway, e.g. when approaching the mouse, and make it bigger in order to ease the interaction.

On the negative side, about half of the users expressed concerns about security issues. Indeed, while many users perceived very positively the simplicity of the interactions (no installation, configuration or log-in) they also perceived this fact as raising a number of security issues. As a consequence, eleven of our twenty users said they would not use the application (or some of the provided services) for confidential or personal data. They suggested automatic authentication and encryption techniques to solve these concerns (betraying a Computer Science background for most of the users).

Eventually, for most users the critical mass of offered services and places offering them was one of the most important points towards user acceptance. As such they suggested implementing providers for multimedia players, service points, file servers, shared calendars and blackboards, scanners, mobile phones and other mobile devices in environments such as business offices, universities, schools and other public facilities.

5. CONCLUSION

The presented proof-of-concept prototype, and its exposure to users, highlight integration of access to pervasive services in mobile user devices in a manner that helps users understand what services are available, and that allows them to interact with services through intuitive techniques. Users experience the provided interface functionality as, in principle, easy to understand and use. However it is also important to understand the limitations of the study carried out so far, in the small number of services integrated and the replacement of real sensor data with Wizard of Oz control. Key questions in further development of the concept are the impact of increased number of services on usability of the interface, the integration with network-level discovery mechanisms, and the impact of using a real sensor system for real-time tracking of spatial relationships between user device and services.

6. ACKNOWLEDGMENTS

This research is part of the Relate project, supported by the Commission of the European Union (Project No. 013790, FP6 IST Programme, RELATE).

7. REFERENCES

[1] G. W. Fitzmaurice. Situated information spaces and spatially aware palmtop computers. *Commun. ACM*, 36(7):39–49, 1993.

[2] F. Hupfeld and M. Beigl. Spatially aware local communication in the raum system. In *IDMS '00: Proceedings of the 7th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services*, pages 285–296, London, UK, 2000. Springer-Verlag.

[3] N. Kohtake, T. Iwamoto, G. Suzuki, D. M. Shun Aoki, T. Kouda, K. Takashio, and H. Tokuda. u-Photo: A Snapshot-based Interaction Technique for Ubiquitous Embedded Information. In *Video Proceedings of the Second International Conference on Pervasive Computing*, 2004.

[4] G. Kortuem, C. Kray, and H. Gellersen. Sensing and visualizing spatial relations of mobile devices. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 93–102, New York, NY, USA, 2005. ACM Press.

[5] M. W. Newman, J. Z. Sedivy, C. M. Neuwirth, W. K. Edwards, J. I. Hong, S. Izadi, K. Marcelo, T. F. Smith, J. Sedivy, and M. Newman. Designing for serendipity: supporting end-user configuration of ubiquitous computing environments. In *DIS '02: Proceedings of the conference on Designing interactive systems*, pages 147–156, New York, NY, USA, 2002. ACM Press.

[6] S. Ponnekanti, B. Lee, A. Fox, P. Hanrahan, and T. Winograd. ICrafter: A Service Framework for Ubiquitous Computing Environments. In *UbiComp '01: Proceedings of the 3rd international conference on Ubiquitous Computing*, pages 56–75, London, UK, 2001. Springer-Verlag.

[7] J. Rekimoto and K. Nagao. The world through the computer: Computer augmented interaction with real world environments. In *ACM Symposium on User Interface Software and Technology*, pages 29–36, 1995.

[8] E. Rukzio, A. Schmidt, and H. Hussmann. Physical posters as gateways to context-aware services for mobile devices. In *Proc. 6th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004)*, 2004.

[9] M. Weiser. The computer for the twenty-first century. *Scientific American*, pages 94–10, 1991.