Supporting Mobile Service Usage through Physical Mobile Interaction

Gregor Broll[1], Sven Siorpaes[1], Enrico Rukzio[2], Massimo Paolucci[3], John Hamard[3], Matthias Wagner[3], Albrecht Schmidt[4]

[1] *Media Informatics Group, University of Munich, Germany*
gregor.broll@ifi.lmu.de, sven@hcilab.org

[2] *Computing Department, Lancaster University, UK*
rukzio@comp.lancs.ac.uk

[3] *DoCoMo Euro-Labs, Germany*
{paolucci, hamard, wagner}@docomolab-euro.com

[4] *Fraunhofer IAIS, Sankt Augustin and B-IT, University of Bonn, Germany*
albrecht.schmidt@ifi.lmu.de

Abstract

Although mobile services can be used ubiquitously, their employment and the interaction with them are still restricted by the constraints of mobile devices. In order to facilitate and leverage mobile interaction with services, we present a generic framework that combines Semantic Web Service technology and Physical Mobile Interaction. This interaction paradigm uses mobile devices to extract information from augmented physical objects and use it for a more intuitive and convenient invocation of associated services. For that purpose, the presented framework exploits Web Service descriptions for the automatic and dynamic generation of customizable user interfaces that support and facilitate Physical Mobile Interaction. This generic approach to mobile interaction with services through the interaction with physical objects promises to meet the complementary development of the Internet of Things. A user study with a prototype application for mobile ticketing confirms our concept and shows its limits.

1. Introduction

The usage of Web Services in the mobile domain is still not as advanced, widespread and established as in Desktop Computing. Despite the technical progress in Mobile Computing, most of its devices – particularly PDAs and mobile phones - only provide inadequate means for the interaction with Web Services and the presentation of their contents, which constrains the mobile usage of Web Services and the usability of applications based on them. Mobile service interaction has to rely on built-in browsers or proprietary service-clients and becomes tedious, intricate and inflexible as it suffers from small screens, fiddly keys and joysticks as well as narrow and glutted menus. This adds to the general problem of adapting mobile application interfaces to different devices, platforms and their individual properties and constraints.

In this context, we present a generic approach that takes advantage of *Physical Mobile Interaction* [15] as a means for supporting and facilitating mobile interaction with services through the interaction with physical objects.

Physical Mobile Interaction is an interaction paradigm that is based on two recent developments in Mobile Computing: On the one hand, physical objects can be increasingly augmented and associated with digital information and on the other hand, mobile devices provide increasing capabilities to ubiquitously acquire and process this information. These developments converge into Physical Mobile Interaction which uses mobile devices to extract information from augmented physical objects and to apply it for a more intuitive and convenient interaction with associated services. Instead of browsing nested and glutted mobile application menus, users can simply touch or point at items or information they want to select or interact with. They can take pictures of visual markers [13] in magazines or on posters and use this information for the automatic invocation of associated services [14]. In combination with wireless technologies like RFID [16] or NFC [10], Physical Mobile Interaction is increasingly gaining importance: It reduces mobile payment, identification or access control to simply swiping a mobile phone over a reader. NTT DoCoMo's i-mode FeliCa service for example combines mobile phones with built-in NFC-chips and a service framework based on i-mode [3].

In our approach, we apply the Physical Mobile Interaction paradigm to mobile service interaction in order to transfer the simplicity of interacting with physical objects to the interaction with associated services and thus make it more convenient and intuitive. In the process, we shift the focus of interaction by pushing features and options off mobile phone menus, mapping them to real world objects (e.g. service options on a poster) and thus turning them into rich ubiquitous interfaces for new and more complex Physical Mobile Interaction techniques.

In order to combine and leverage Physical Mobile Interaction and Web Services for their mutual benefit, we developed a generic framework that exploits Semantic Web Service descriptions for the dynamic and automatic generation of adaptable interfaces that support and facilitate Physical Mobile Interaction. This approach provides a powerful foundation for realizing more complex and flexible interaction techniques that benefit from the interoperability, extendibility and expressiveness of Semantic Web Service technologies. On the other hand, mobile services benefit from the seamless integration with Physical Mobile Interaction. It provides a more natural and intuitive way of interacting with services, which could leverage their usage, dissemination and availability. Our approach merges both technologies into an independent service framework that can be adapted to and used across different services, interaction techniques, platforms and devices.

The next chapter provides an overview of different work that is related to our approach. Chapter 3 introduces the architecture of our framework, while chapters 4 and 5 give details about the generation of interfaces for mobile service interaction and the implementation of Physical Mobile Interaction. Chapter 6 presents a prototype application for mobile ticketing as well as a user study that was conducted in order to test and evaluate our approach. The conclusion in chapter 7 summarizes our work and gives an outlook to future issues.

2. Background and Related Work

Our research is inspired and outlined by similarities and differences with related work in various areas, especially about the description and automatic generation of interfaces as well as Physical Mobile Interaction.

User interface description languages (UIDL) provide the basis for interface generation. Concrete UIDLs such as XUL [8] comprise a vast set of detailed widgets for building rich application interfaces, but are less suitable for the generation of interfaces from Web Service descriptions. More abstract UIDLs like UIML [1] on the other hand have been designed to describe and create generic user interfaces for applications on different devices and platforms. Due to differences in the vocabularies of interface rendering engines, single UIML descriptions still have to be tailored to their syntax and can't be used for the derivation of multiple interfaces for different target platforms. Like most UIDLs, UIML neither supports the generation of mobile user interfaces nor connections to Web Services in particular.

The Personal Universal Controller (PUC) [9] utilizes mobile devices as "remote controls" for the interaction with common electrical appliances (e.g. stereo sets or VCRs) which provide abstract descriptions of their functionalities. Mobile devices can download these specifications and use them for the automatic generation of user interfaces for controlling the corresponding appliances. While this approach to generating mobile interfaces is similar to ours, it does not support the interaction with mobile services.

In [5], Khushraj and Lassila use descriptions from the OWL-S Service Profile and Process Model of Semantic Web Services as the basis for the generation of dynamic form-based user interfaces. As some interface properties can not be derived from these descriptions, they are extended with additional OWL-S annotations about labels, preferred widget types or the grouping of fields and sub-fields. Although this procedure only supports the creation and customization of simple form-based interfaces, it inspires our own approach which extends the presented idea to interfaces for different platforms and to the generic support of Physical Mobile Interaction.

Riekki, Salminen and Alakarppa [12] developed a framework for requesting services by touching RFID tags, including a middleware and tags with different functions that is very close to our own approach although we put a stronger focus on Web Services, generic interface generation and Physical Mobile Interaction.

The development of Physical Mobile Interaction is met and advanced by the dissemination of another related technology: The *Internet of Things* [7], in which everyday objects are uniquely identified through wireless markers and have individual network references for easier recognition, identification and monitoring. Physical Mobile Interaction could establish itself as a natural complement to the Internet of Things, as it facilitates the interaction of its objects with associated information and services.

3. Framework Architecture

The approach to mobile interaction with services through the interaction with physical objects is reflected in the architecture of our framework as it integrates Web Services and Physical Mobile Interaction into a coherent system. Different requirements influenced its design:

- Abstract description and extension of services in order to generate generic, customizable interfaces
- Reuse of single service descriptions for generating multiple user interfaces
- Automatic and dynamic adaptation of interfaces to different mobile devices, target platforms, user profiles and interaction designs instead of being dependent on them
- Strong technical and functional relation between physical objects and mobile services
- Support for different interaction techniques

These and other requirements were considered during the development of the framework, whose architecture is divided into 3 major components (Figure 1): The *Physical Mobile Interaction Domain* comprises mobile devices and client applications that interact with physical objects. These objects are augmented with different technologies (e.g. NFC tags, visual markers or Bluetooth) in order to provide information that can be used as input for the invocation of services that are associated with them.

These services are part of the *Web Service Domain* and represent the backend logic in the architecture. They are exchangeable and can be easily replaced in order to provide Physical Mobile Interaction for other Web Services, e.g. offered by Amazon. The descriptions of service functionalities, along with different extensions and annotations provide the basis for the generation of interfaces for Physical Mobile Interaction.

The *Interaction Proxy* mediates between Web Services and client applications as it bridges the gap between both domains. In order to increase the efficiency of the interface generation process, it is intended to adopt and centralize common functionalities that can be outsourced from the Web Service Domain (e.g. interface generation) and constrained mobile devices (e.g. resource-demanding transformation processes). That way, the Interaction Proxy ties the two separate domains together while retaining their independence from each other. It also keeps the framework generic enough for the future integration and support of additional Web Services, client technologies and physical objects. Depending on the technical performance of mobile client devices, the Interaction Proxy can be either partially or fully implemented and executed on these devices or on a separate server.

Consequently, the Interaction Proxy assists the *Universal Client* application in the Physical Mobile Interaction Domain and provides complete interfaces (e.g. HTML) or compact interface descriptions for client side rendering. The Universal Client is an

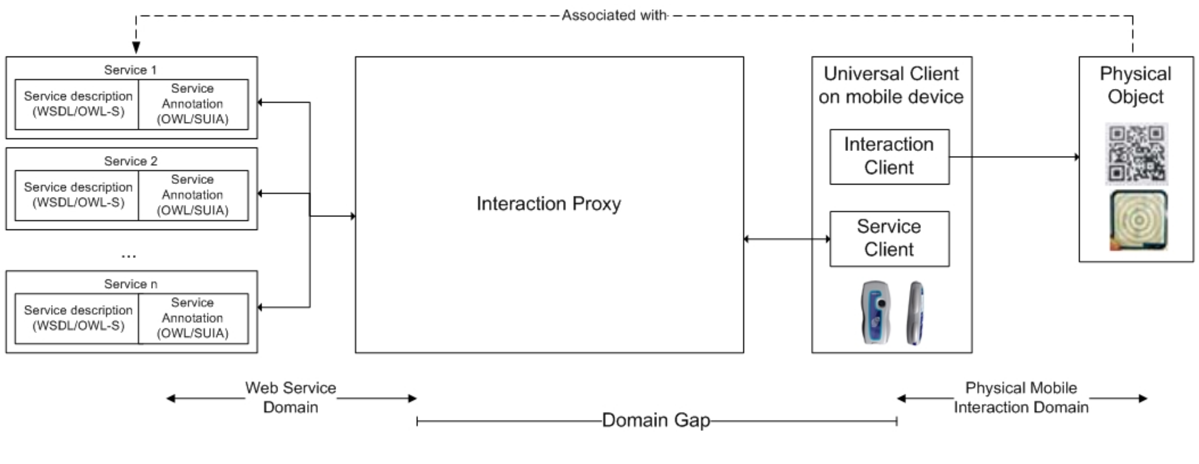

Associated with
Service 1
Service description (WSDL/OWL-S)
Service Annotation (OWL/SUIA)
Service 2
Service description (WSDL/OWL-S)
Service Annotation (OWL/SUIA)
...
Service n
Service description (WSDL/OWL-S)
Service Annotation (OWL/SUIA)
Interaction Proxy
Universal Client on mobile device
Interaction Client
Service Client
Physical Object
Web Service Domain
Domain Gap
Physical Mobile Interaction Domain

Figure 1. Architecture of the generic service framework, that takes advantage of Physical Mobile Interaction for the invocation of Web Services

application on a mobile device that acts as a mediator between the Interaction Proxy and different physical objects. Its *Service Client*-component provides an interface for the communication with the Interaction Proxy while the *Interaction Client*-component manages and abstracts different technologies for Physical Mobile Interaction with everyday objects. In addition, the Universal Client represents the generic application logic that renders, displays and uses interfaces for this interaction and the mobile invocation of Web Services.

4. Service Annotation and Interface Generation

The main requirement of the presented framework is providing the means to leverage the automatic generation of interfaces for physical, mobile service interaction and their dynamic adaptation to different contextual constraints. In order to meet this and other requirements (see chapter 3), the framework exploits the expressiveness and flexibility of (Semantic) Web Service technology for the embodiment of service descriptions and different service extensions (see Figure 2). The latter are extendable, reusable ontologies that help implement general, independent concepts and bridge the gap between Web Services and Physical Mobile Interaction.

The Web Services that are part of the framework use the Web Services Description Language (WSDL) [4] for the deployment and invocation of Web Services as well as for the definition of service parameters and message formats. Since the merely functional WSDL service descriptions do not provide sufficient information for generating user interfaces, the framework uses them as the grounding for semantic OWL-S [6] service descriptions (see Figure 2). These OWL-S service ontologies build the basis for the interface generation process as they provide a higher level of expressiveness and allow the specification of more complex interaction sequences.

In order to provide additional information for the generation and rendering of user interfaces that go beyond the still functional OWL-S service descriptions, the framework extends them with the *Service User Interface Annotation (SUIA)* ontology (see Figure 2). Its main class *AbstractUIMapModel* (see Figure 3) serves as a collection bag for several parameter mappings represented by instances of the class *ParamterMap* which are attached by the property *hasParameterMap*. The information encapsulated in a ParameterMap complements the description of a specific input or output parameter in the OWL-S service ontology which is indicated by the object property *hasServiceParameterRef*. The remaining properties in the ParameterMap express information that is needed for rendering the interfaces and increasing their usability:

- *hasAbstractWidgetType*: abstract specification of the type of widget that the target platform should map to a parameter
- *hasLabel*: readable label to increase the expressiveness and usability of interfaces
- *hasDescription*: additional description of a parameter
- *hasImageRef*: optional definition of a URL to associate an image with the parameter
- *hasParameterValueSet*: predefined sets of values that are valid for an input parameter type

The ParameterMap-property hasAbstractWidget-Type refers to the *Abstract Widget Type Model* that describes different abstract widget types in a general way independent from target device characteristics

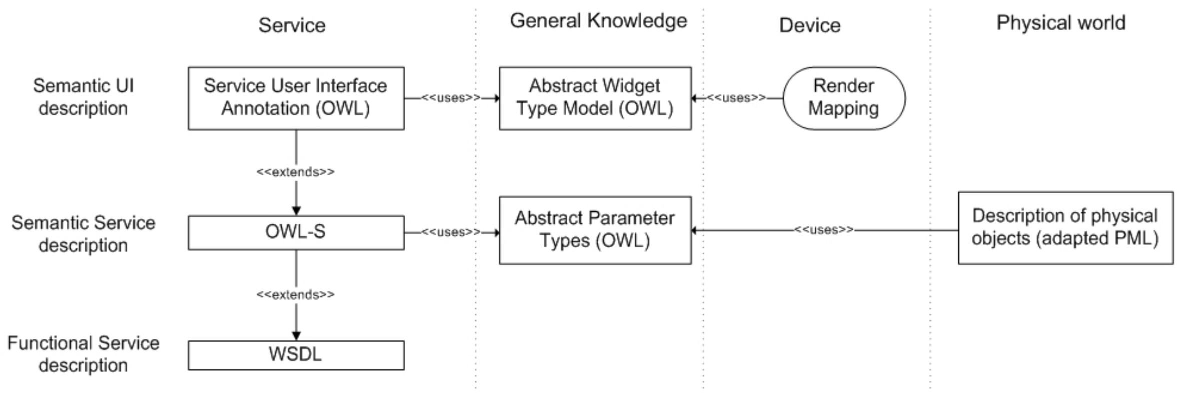

Service
General Knowledge
Device
Physical world
Semantic UI description
Service User Interface Annotation (OWL)
<<uses>>
Abstract Widget Type Model (OWL)
<<uses>>
Render Mapping
<<extends>>
Semantic Service description
OWL-S
<<uses>>
Abstract Parameter Types (OWL)
<<uses>>
Description of physical objects (adapted PML)
<<extends>>
Functional Service description
WSDL

Figure 2. Service descriptions and extensions as the basis for interface generation, customization and rendering

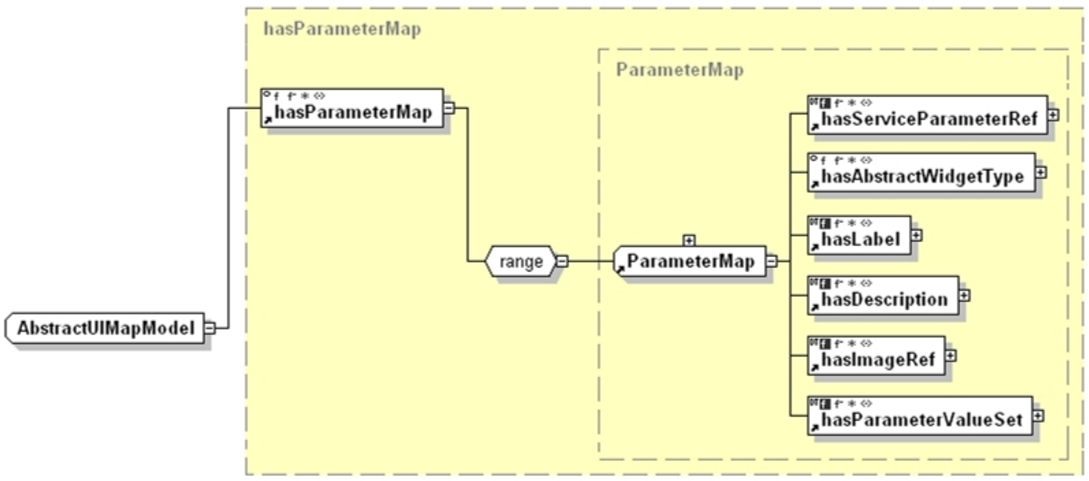


Figure 3. The AbstractUIMapModel of the Service User Interface Annotation ontology containing information for rendering usable interfaces

such as rendering platform and language specific user interface concepts. Its information is interpreted by the interface rendering mechanism of the Universal Client and mapped to the concrete user interface widgets of a target platform. Figure 4 depicts the hierarchy and the different widget types of the Abstract Widget Type Model that is specified in a self-contained OWL-S ontology and shared as a general concept between the domains of Web Services and Physical Mobile Interaction (see Figure 2).

The Abstract Widget Type Model represents the most common widget concepts in user interfaces and can be extended with additional components. The general class *WidgetType* distinguishes between subclasses for inputs (*InputWidgetType*) and outputs (*OutputWidgetType*) depending on the type of service parameter to be augmented. Depending on the target platform and programming language, these generic interface identifiers serve as hints for their rendering as concrete interface widgets. Different applications might display them as input fields, checkboxes, option menus, radio buttons, hidden fields or any widget that fits the denounced generic widget type.

Apart from the service extensions, OWL-S ontologies are also used for the definition and assignment of *Abstract Parameter Types,* an abstract information typing model within the framework that expresses the functional correlation between physical objects and Web Services (see Figure 2). Just as objects are associated with certain services (see Figure 1), this model types and associates information on these objects (e.g. options on a movie poster) with

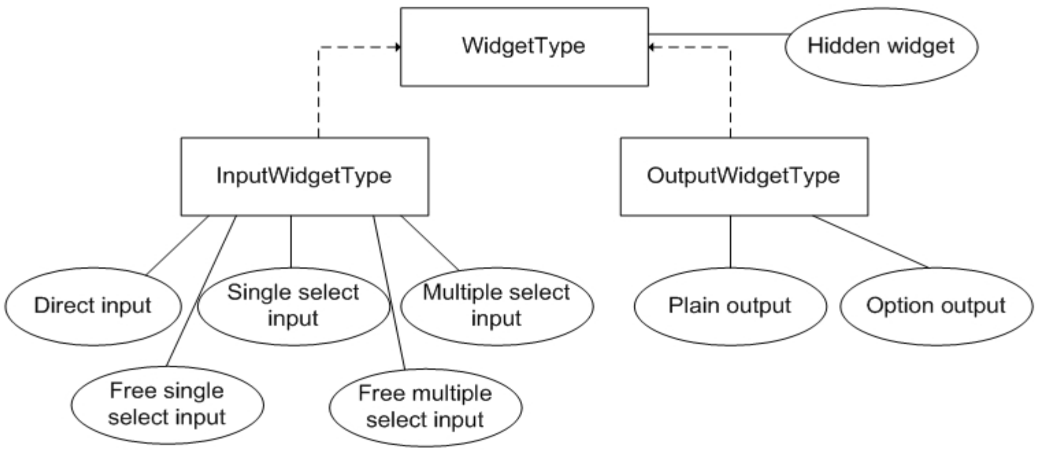


Figure 4. The Abstract Widget Type Model for specifying the rendering of individual interface widgets

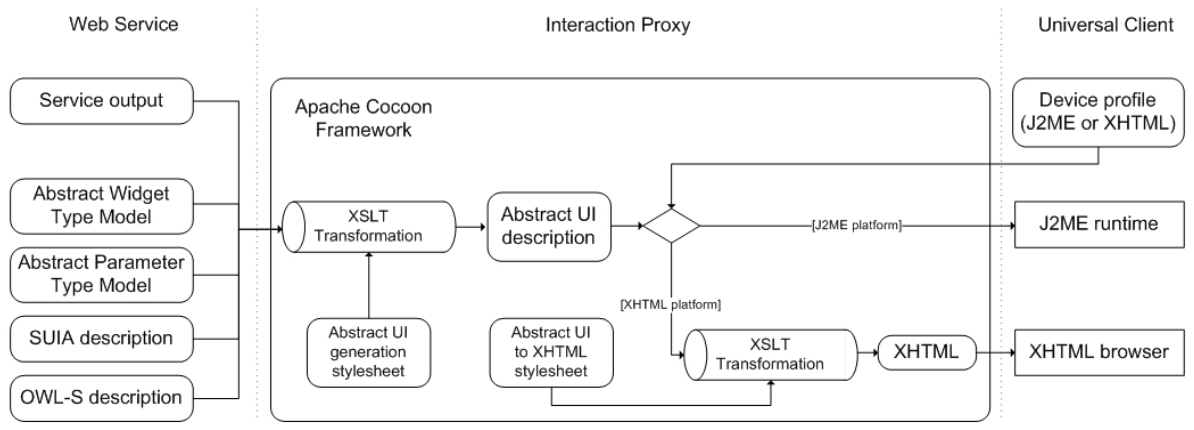


Figure 5. The multichannel interface generation and transformation process

equally typed input parameters of the corresponding services (e.g. for purchasing movie tickets). The *Abstract Parameter Type Model* is another general concept in the architecture that is defined independently from its other components in order to stay extendible and reusable.

Figure 5 provides an overview of the interface generation process within the Interaction Proxy. In order to be efficient and flexible, it tries to accomplish the generic support for different target platforms, interaction techniques, user preferences and other contextual constraints with the least effort and greatest reuse of resources. The first step is the composition of the basic OWL-S Web Service descriptions, their extensions and other ontologies (SUIA, Abstract Parameter Type Model and Abstract Widget Type Model) into the *Abstract User Interface (UI) Description* that serves as the basis for the further creation and customization of mobile service interfaces. Additional input may be provided by the service implementation itself, e.g. the output of a previous interaction step.

The Interaction Proxy gathers these descriptions and forwards them to an integrated instance of Cocoon [2], an XML-framework for multi-channel publishing that implements the generation and transformation of

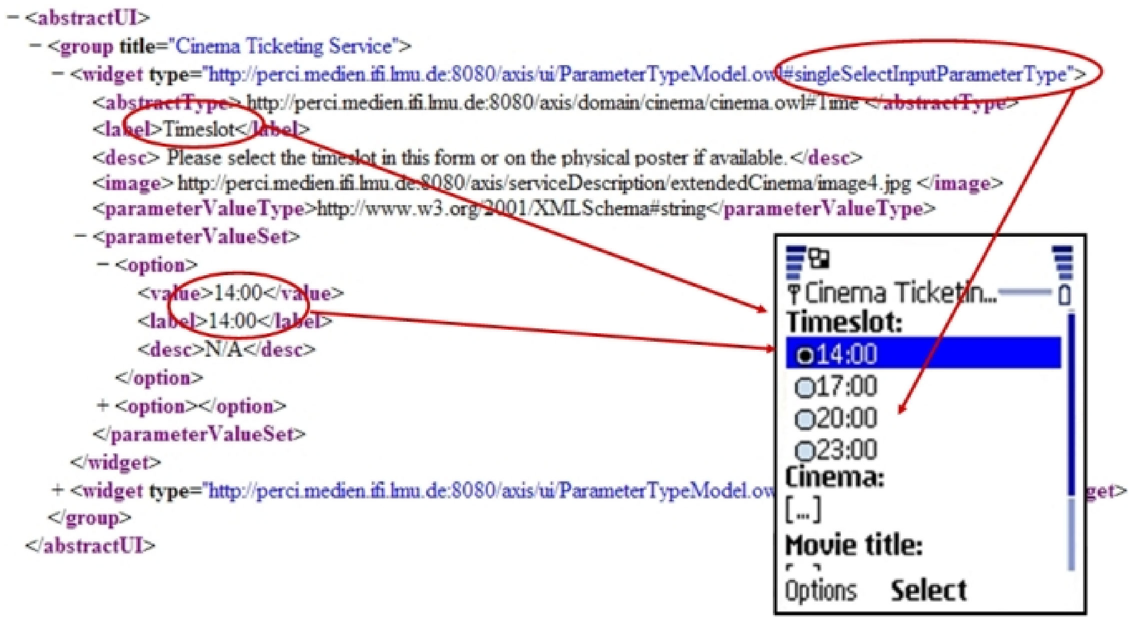

```
- <abstractUI>
  - <group title="Cinema Ticketing Service">
    - <widget type="http://perci.medien.ifi.lmu.de:8080/axis/ui/ParameterTypeModel.owl#singleSelectInputParameterType">
        <abstractType>http://perci.medien.ifi.lmu.de:8080/axis/domain/cinema/cinema.owl#Time </abstractType>
        <label>Timeslot</label>
        <desc> Please select the timeslot in this form or on the physical poster if available.</desc>
        <image>http://perci.medien.ifi.lmu.de:8080/axis/serviceDescription/extendedCinema/image4.jpg </image>
        <parameterValueType>http://www.w3.org/2001/XMLSchema#string</parameterValueType>
      - <parameterValueSet>
        - <option>
            <value>14:00</value>
            <label>14:00</label>
            <desc>N/A</desc>
          </option>
        + <option></option>
        </parameterValueSet>
      </widget>
    + <widget type="http://perci.medien.ifi.lmu.de:8080/axis/ui/ParameterTypeModel.ow                    get>
    </group>
  </abstractUI>
```

Figure 6. An Abstract User Interface Description and how its different elements are rendered into a concrete interface

interfaces. Cocoon aggregates the different descriptions, applies appropriate XSLT-stylesheets and derives the Abstract UI Description according to its transformation rules (see Figure 5). This new and abstract service description is a summary of the previous descriptions and contains all necessary information for further interface generation while being more concise and easier to interpret (see Figure 6). Basically, it lists abstract descriptions of widgets that collect input values for the different parameters of a service.

After the Abstract UI Description has been generated, it is rendered into a concrete user interface by either the Interaction Proxy itself or the Universal Client, depending on the interaction design, supported platforms, interaction techniques or user preferences. The current implementation of the Interaction Proxy supports HTML and Java ME interfaces. It recognizes information about the target platform from client requests and has Cocoon use this information for choosing different branches in its multi-channel publishing process (see Figure 5). If the system recognizes a HTML user agent header, another XSLT-transformation translates the Abstract UI Description into a HTML-document that is returned to the Universal Client and displayed by its HTML-browser. In case the Universal Client recognizes appointed Http-request properties that indicate the usage of a Java ME client-midlet, the Abstract UI Description itself is returned and rendered by the Java ME runtime environment of the Universal Client according to its own rules. Figure 6 shows how the different information in an Abstract UI Description is rendered into different parts of a Java ME interface.

5. Supporting Physical Mobile Interaction

As the last major component in the framework, the Universal Client embodies the concept of a generic client platform for Physical Mobile Interaction with services. Its actual implementation and range of capabilities are dependent on the platform (e.g. Symbian, Windows Mobile, EPOC, etc.), supported technologies (e.g. Bluetooth, NFC, GPS, camera, etc.) and programming languages (e.g. HTML, Java ME, etc.) provided by different mobile devices. Therefore, Universal Clients for different platforms vary in the complexity of their functionalities and can range from a simple (mobile) Web-browser to a specialized client for different interaction techniques. According to their specific implementation, they either receive complete interfaces (e.g. HTML) from the Interaction Proxy and display them for service interaction or generate them from the Abstract UI Description themselves. Either way, these interfaces support and visualize the mobile interaction between the Universal Client and physical objects.

Depending on its technical abilities, a Universal Client implements different Physical Mobile Interaction techniques and manages them through its Interaction Client-component. That way, the Universal Client can interact with physical objects, collect their information and use it for the invocation of services in a generic way while staying independent from any of them. Examples for interaction techniques and the technologies behind them are:

- For *Touching* (Figure 7a) physical objects are augmented with NFC or RFID tags that store information, e.g. for identification or service invocation. Users select these tags by touching and reading them e.g. with their mobile phones.
- For *Pointing* (Figure 7b) physical objects are augmented with visual markers that can be captured with mobile phone cameras and interpreted by image recognition algorithms.

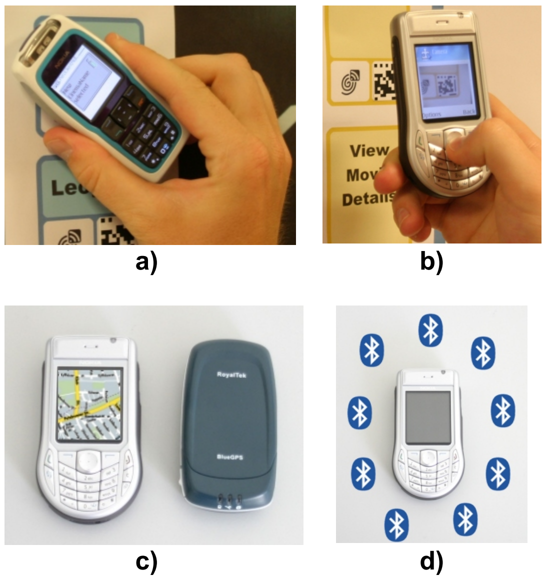


Figure 7. Different techniques for the mobile interaction with physical objects

- *Location Based Object Selection* (Figure 7c) uses external GPS devices via Bluetooth in order to measure the proximity to physical objects and to interact with them accordingly, e.g. when a user moves within the vicinity of an object.
- *Scanning* (Figure 7d) uses the Bluetooth capabilities of mobile phones in order to search

for, connect to and interact with other devices and augmented objects in their vicinity.

Since many mobile devices don't possess the technical equipment for the previous interaction techniques, *Direct Input* can also be considered for Physical Mobile Interaction. It simply uses the standard widgets of its client platform in order to provide an application with information.

The interfaces that are generated within the framework support Physical Mobile Interaction to an extent that is determined by the technological equipment of the corresponding client platform. Devices with HTML-browsers can only support the direct input of values through HTML-forms. Java ME-enabled mobile phones on the other hand can implement several of the presented interaction techniques, provided that they possess the necessary hardware, e.g. digital cameras or built-in NFC-readers.

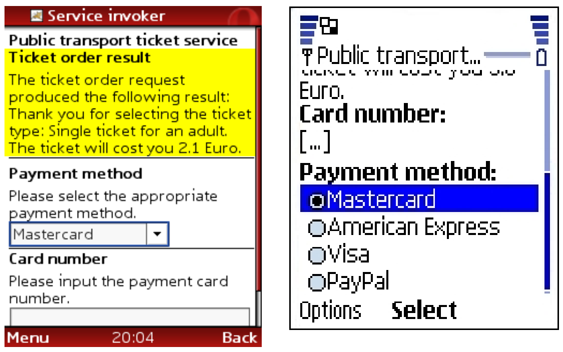
6. Prototype Implementation and Evaluation

In order to visualize, test and evaluate our approach to mobile service interaction, we developed two use-case-scenarios for mobile ticketing using augmented posters and implemented a Universal Client prototype for Physical Mobile Interaction with the corresponding services.

The current implementation of the framework returns complete HTML-interfaces (see Figure 8a) or Abstract UI Descriptions for individual interface generation. Any (mobile) device with an HTML-browser can use it as a Universal Client to display the HTML-interfaces. Although mobile service interaction benefits from the familiarity of using HTML and its wide dissemination, common browsers are hard to extend in order to support any Physical Mobile Interaction techniques other than Direct Input through HTML's form-based input-widgets.

In order to take more efficient advantage of Physical Mobile Interaction, a Java ME prototype client was developed, that implements Touching through NFC, Pointing through visual marker recognition and Direct Input through standard input-widgets. The prototype client implementation uses the *Physical Mobile Interaction Framework (PMIF)* [15] for the generic integration and easy exchange of different interaction techniques.

During the interface generation process, the Service Client-component of the prototype receives the Abstract UI Description of a service from the Interaction Proxy. The client derives a suitable Java ME-widget from each abstract widget description, whose assignment is suggested in the Abstract UI Description (see Figure 6) and is ultimately decided by the interaction design of the prototype implementation (see Figure 8b). That way, an abstract widget description denoted as *SingleSelectInput* could be rendered as a list of radio buttons for Direct Input or as a read-only text-field that forces users to use Touching or Pointing in order to pick parameter-values from physical objects and place them into the interface.


Service invoker

Public transport ticket service

Ticket order result

The ticket order request produced the following result: Thank you for selecting the ticket type: Single ticket for an adult. The ticket will cost you 2.1 Euro.

Payment method

Please select the appropriate payment method.

Mastercard ▾

Card number

Please input the payment card number.

Menu 20:04 Back

Public transport...

Euro.

Card number:

[...]

Payment method:

● Mastercard
○ American Express
○ Visa
○ PayPal

Options **Select**

a) b)

Figure 8. Interfaces for service interaction through different mobile clients

During Physical Mobile Interaction, the content of the rendered widgets is updated with the acquired values, which have been specified using the Abstract Parameter Type Model. After all necessary values have been collected and displayed in the interface for visual feedback, they are extracted from their widgets and serve as request-parameters for the invocation of the corresponding service. Both HTML- and Java-interfaces use the Abstract Parameter Type Model in order to associate the information acquired from physical objects with the corresponding interface widget and thus service input parameters of the same type.

Apart from the Universal Clients for HTML- and Java ME–interfaces, the prototype implementation also comprises 2 posters that were used as physical objects in the context of 2 use-case scenarios for mobile ticketing. The first poster allows users to purchase movie tickets and offers appropriate options (movie title, cinema name, number of tickets and preferred timeslot) together with a selection of values (see Figure 9). The second poster implements a simplified way to buy tickets for a public transportation system (see Figure 10). Instead of having to understand a complicated ticketing system, inexperienced users only have to select options for the station they want to start their journey from, their destination, the number of passengers as well as the duration of the ticket in order to have appropriate tickets suggested.

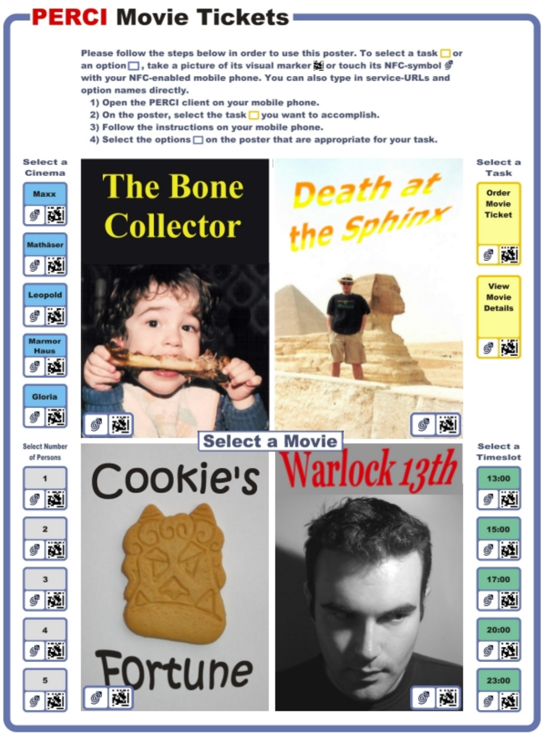

PERCI Movie Tickets
Please follow the steps below in order to use this poster. To select a task or an option, take a picture of its visual marker or touch its NFC-symbol with your NFC-enabled mobile phone. You can also type in service-URLs and option names directly.
1) Open the PERCI client on your mobile phone.
2) On the poster, select the task you want to accomplish.
3) Follow the instructions on your mobile phone.
4) Select the options on the poster that are appropriate for your task.
Select a Cinema
Maxx
Mathäser
Leopold
Marmor Haus
Gloria
The Bone Collector
Death at the Sphinx
Select a Task
Order Movie Ticket
View Movie Details
Select a Movie
Select Number of Persons
1
2
3
4
5
Cookie's Fortune
Warlock 13th
Select a Timeslot
13:00
15:00
17:00
20:00
23:00

Figure 9. Use case poster for mobile ticketing (cinema)

The posters and the prototype client implementation use different action- and parameter-tags which are logically mapped to different services and their parameters. Action-tags contain the URLs of different services while parameter-tags provide parameter-values for their invocation. That way, a poster can support the invocation of different services that use the same (sub)set of parameters. Although both tags are independent from each other, the contents of parameter-tags can be associated with the input-parameters of services through the commonly applied Abstract Parameter Type Model. On the posters, each tag is mapped to its own option and stores its information through the augmentation with NFC-tags, visual markers and a human readable identifier.

In order to evaluate our approach, we conducted 2 user studies, for which subjects (10 per study) had to use the prototype client application and interact with the posters. While the first study only used paper-prototyping to estimate the overall concept and acceptance of our approach, the second study was conducted with the implemented framework and evaluated Physical Mobile Interaction using a mobile HTML-browser and the Java ME Universal Client-implementation.

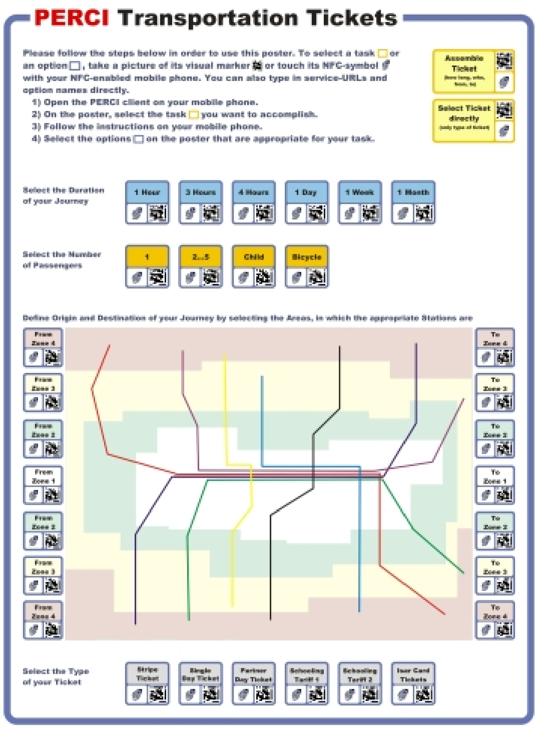

PERCI Transportation Tickets
Please follow the steps below in order to use this poster. To select a task or an option, take a picture of its visual marker or touch its NFC-symbol with your NFC-enabled mobile phone. You can also type in service-URLs and option names directly.
1) Open the PERCI client on your mobile phone.
2) On the poster, select the task you want to accomplish.
3) Follow the instructions on your mobile phone.
4) Select the options on the poster that are appropriate for your task.
Assemble Ticket
Select Ticket directly
Select the Duration of your Journey
1 Hour
3 Hours
4 Hours
1 Day
1 Week
1 Month
Select the Number of Passengers
1
2...5
Child
Bicycle
Define Origin and Destination of your Journey by selecting the Areas, in which the appropriate Stations are
From Zone 4
From Zone 3
From Zone 2
From Zone 1
From Zone 2
From Zone 3
From Zone 4
To Zone 4
To Zone 3
To Zone 2
To Zone 1
To Zone 2
To Zone 3
To Zone 4
Select the Type of your Ticket
Stripe Ticket
Single Day Ticket
Partner Day Ticket
Schooling Tarif 1
Schooling Tarif 2
Isar Card Tickets

Figure 10. Use case poster for mobile ticketing (transportation)

Both user studies confirmed our approach to mobile service interaction through Physical Mobile Interaction: The whole system and especially the usage of Physical Mobile Interaction was generally very well received, despite an initial inhibition level due to the unfamiliar concept of different action- and parameter-tags and the new type of interaction in general. Concerning the different Physical Mobile Interaction techniques, Touching was the clear favorite regarding usability, innovation and reliability. Pointing on the other hand was perceived as the exact opposite: more complicated to use, less reliable and less convenient – disadvantages that can be accredited to the additional effort of using a mobile phone camera. Direct Input benefited from its easy handling and great reliability but was not considered to be innovative or interesting. Despite the generally great acceptance of Physical Mobile Interaction, it also became evident that the potential of Physical Mobile Interaction is not yet fully exploited and needs more support. This showed especially as subjects did not use the poster and the application as intuitively and naturally as expected.

7. Conclusion and Outlook

We presented an approach to make mobile interaction with services easier and more convenient. It dodges the constraints of mobile devices and leverages their otherwise restricted means of interaction. For that purpose we developed a generic framework that combines services and Physical Mobile Interaction for their mutual benefit. Its implementation provides basic support for the dynamic and automatic generation of interfaces from extended service descriptions. Thus it enables and facilitates the development of more complex mobile interactions with physical objects and their associated services. For that purpose, it uses abstract type systems for service parameters and interface widgets in order to link physical objects with services and adapt interfaces to different client platforms while staying independent from them. We conducted two user studies with 10 participants each, to evaluate the implementation of the framework and a mobile client prototype, which confirmed our approach but also showed its limits.

Currently implementing the generic integration of different services and Physical Mobile Interaction techniques, the framework can be extended and improved in several ways: It could adapt interfaces to a wider range of client platforms and add support for additional interaction techniques. The differentiation between action- and parameter-tags could be extended to other, more convenient and intuitive interaction designs. Tags could be used across multiple posters or serve as input for search engine queries that point to services that can be invoked with the poster's tags. Authoring tools for the framework could abstract and facilitate the different steps that are needed for setting up services and connecting them to physical objects. Finally, all these efforts could evolve Physical Mobile Interaction for better integration with other, compatible technologies like the Internet of Things.

8. Acknowledgement

The presented approach to the combination of Web Services and Physical Mobile Interaction was developed within the PERCI-project (PERvasive ServiCe Interaction) [11], a collaboration of the University of Munich and NTT DoCoMo Euro-Labs that is funded by the latter.

9. References

[1] Abrams, M. and C. Phanouriou. "UIML: An XML Language for Building Device-Independent User Interfaces". XML '99, Philadelphia, 1999.
[2] Apache Cocoon Project. http://cocoon.apache.org/.
[3] Boyd, J. "Here Comes The Wallet Phone". IEEE Spectrum Online. Nov 2005. www.spectrum.ieee.org/nov05/2150
[4] Christensen, E., F. Curbera, G. Meredith, and S. Weerawarana. "Web Service Description Lanaguage (WSDL) Homepage". 2001. http://www.w3.org/TR/wsdl.
[5] Khushraj, D. and O. Lassila: "Ontological Approach to Generating Personalized User Interfaces for Web Services". International Semantic Web Conference 2005, 2005.
[6] Martin, D., M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. "OWL-S: Semantic Markup for Web Services". 19.06.2006. http://www.daml.org/services/owl-s/1.1/overview/.
[7] Meloan, S. "Toward a Global 'Internet of Things'". 11.11.2003. http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/.
[8] Mozilla.org. XML User Interface Language (XUL). 03.07.2006. http://www.mozilla.org/projects/xul/.
[9] Nichols, J. "Automatically Generating User Interfaces for Appliances". In: UIST 2004 Conference Companion, Santa Fe, NM, 2004.
[10] NFC Forum. http://www.nfc-forum.org/.
[11] Perci homepage. http://www.hcilab.org/projects/perci/
[12] Riekki, J., T. Salminen, and I. Alakarppa. "Requesting Pervasive Services by Touching RFID Tags". IEEE Pervasive Computing 5, Jan. 2006.
[13] Rohs, M. and B. Gfeller. "Using Camera-Equipped Mobile Phones for Interacting with Real-World Objects". In: Ferscha, A., H. Hoertner, and G. Kotsis (Eds.): Advances in Pervasive Computing. Austrian Computer Society (OCG), Vienna, Austria, 2004.
[14] Rukzio, E., A. Schmidt, and H. Hussmann. „Physical Posters as Gateways to Context-aware Services for Mobile Devices". Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004), English Lake District, UK, 2004.
[15] Rukzio, E., S. Wetzstein, and A. Schmidt. „A Framework for Mobile Interactions with the Physical World". Wireless Personal Multimedia Communication (WPMC'05). Aalborg, Denmark, 2005.
[16] Want, R. "An Introduction to RFID Technology". IEEE Pervasive Computing, vol. 5, pp. 25-33, 2006.