Supporting Mobile Service Interaction through Semantic Service Description Annotation and Automatic Interface Generation

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

[1] Media Informatics Group, University of Munich, Germany
{gregor, sven, enrico, albrecht}@hcilab.org
[2] DoCoMo Euro-Labs, Germany
{paolucci, hamard, wagner}@docomolab-euro.com

Abstract. One of the current challenges in Mobile Computing is bringing services directly to mobile users, which is handicapped by two major hurdles: Phones and services do not interoperate as smoothly as they should and the delivered services have to be adapted to a wide range of different mobile client platforms. In order to address these problems we present a service framework that extends Semantic Web Service descriptions with abstract interface annotations and uses them for the automatic generation of adapted user interfaces. These interfaces support and facilitate the mobile interaction with physical objects and thus the interaction with associated Semantic Web Services. The focus of this paper lies on the service description annotations based on OWL-S. Furthermore we show how these extensions can be used for the generation of a compact and abstract interface description as the basis for the rendering of Java ME and XHTML - based interfaces. In order to motivate our approach and confirm its concept, we developed two prototypes for mobile ticketing that are based on the presented system.

1 Introduction

Web Service technology provides powerful means to deliver information through the Internet, and Web Service languages greatly facilitate the description of networked applications as well as their interoperation with clients. Due to these characteristics, Semantic Web Service technology is probably the most promising candidate for connecting mobile devices to pervasive services. In an ideal world, a mobile user may enter a new environment, download the semantic descriptions of the services that are available and automatically connect to them in order to gather information about the environment or to access new services.

Still, despite the progress that has been registered in Web Service computing and despite the fact that an increasing number of mobile devices supports this technology, the mobile usage of Web Services is still rather constrained and not as common as it could be. We believe that this development has several reasons: In order to be successful, Web Services have to guarantee full automatic interoperation with each other and mobile devices. So far the application of Web Services has been limited to connecting them to the back end of B2B applications, and by and large they are not

exposed to human users. Additionally, the provision and use of Web Services has to struggle with the wide range of different mobile client platforms, their properties and constraints, concerning e.g. display size, memory, processing power or interaction capabilities.

In order to address and ease these problems that handicap the widespread mobile use of Web Services, we propose a generic approach to exploit Semantic Web Service technology and combine it with Physical Mobile Interaction for their mutual benefit. Physical Mobile Interaction uses different technologies and techniques for the interaction with everyday objects. Thus it provides more intuitive access to information that is associated with these objects and facilitates the interaction with corresponding services. For example, mobile phones can take pictures of visual markers [1] in magazines or on posters and use this information as parameters for the automatic invocation of associated services [2].

The presented approach exploits Semantic Web Service descriptions for the automatic and dynamic generation of adapted interfaces that support and facilitate Physical Mobile Interaction. Semantic Web Service technology helps realising mobile interaction with services as it enhances the interoperability, extensibility, expressiveness and independence of Web Services and their descriptions. These service descriptions can be reused and adapted to different mobile devices, target platforms, user profiles and interaction designs. Thus they provide powerful resources for the development of rich Physical Mobile Interactions.

On the other hand, Web Services can benefit from Physical Mobile Interaction, as it provides a more natural and intuitive way of interacting with them, which could leverage their usage, dissemination and availability. Instead of having to scroll through glutted mobile phone menus on tiny screens, users can simply touch or point at everyday objects and thus interact with associated information and services.

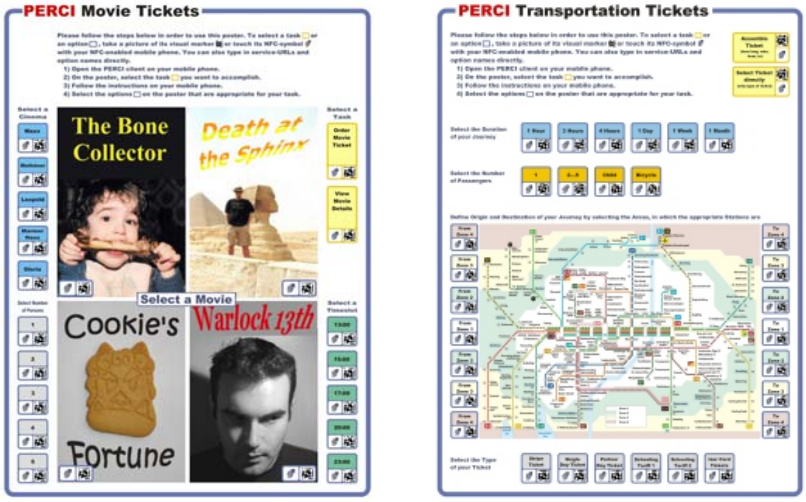
Additionally, Physical Mobile Interaction meets and advances the development of the Internet of Things [3], in which everyday objects receive individual network references through the augmentation and identification with wireless markers. Its standardized infrastructure for identifying, describing and monitoring objects accommodates the association with Web Services and provides a great foundation for the Physical Mobile Interaction with them.

For the focus of this paper, we present a framework that uses interface annotations in order to extend Semantic Web Service descriptions. These enhanced service descriptions serve as the foundation for the automatic generation of adaptable user interfaces that facilitate the mobile interaction with corresponding Web Services.

In the next chapter, we introduce use-case scenarios for mobile ticketing in order to motivate the presented approach to mobile interaction with Web Services. The following chapters give an overview of our framework, deal with the extension of Semantic Web Service descriptions and explain the process of transforming them into customised user interfaces. A chapter on related work compares our approach to other, similar work in the field and the conclusion summarises this paper and gives an outlook to future work.

2 Use Case Scenarios for Mobile Service Interaction

In order to motivate our approach to mobile interaction with Semantic Web Services, we developed two use case scenarios for mobile ticketing. These scenarios realize mobile interaction with Web Services through the interaction with posters that are associated with these services. The first poster can be used to buy movie tickets and offers different appropriate options like movie title, cinema name, number of tickets/persons and timeslot, together with a selection of values (Fig. 1a). The second poster implements a simplified way to buy tickets for a public transportation system (Fig. 1b). Its users only have to select the options and values for the start of their journey, their destination, the number of persons and the duration of the ticket in order to have appropriate tickets suggested. More experienced users can directly select from a list of the most frequently requested tickets.



a) **b)**

Fig. 1. Use-case posters for mobile ticketing

These posters are examples for interfaces that are distributed between mobile phones and physical objects. Mobile client interfaces that have been rendered from Semantic Web Service descriptions guide the interaction with the posters and assist in the invocation of associated Web Service. The parameters for this invocation are provided by the different options on the posters and the values they hold. Instead of being hidden in long, glutted lists and nested menus, these options have been pushed onto the posters and users can select them more conveniently through Physical Mobile Interaction. For this purpose, the posters and the mobile client application support and implement 3 different Physical Mobile Interaction techniques: Users can

select options and their values on the posters by touching them with NFC-enabled mobile phones and reading data stored on NFC-tags attached to the back of the posters (see Fig. 2a), by taking pictures of visual markers and analyzing their encrypted code (see Fig. 2b) or by directly typing the values of options into a form on the mobile device (see Fig. 2c).

a) **b)** **c)**

Fig. 2. Different Physical Mobile Interaction techniques

The posters and the corresponding mobile client application have been evaluated and redesigned in two user studies using different designs for the posters, paper-prototyping for the mobile client as well as implementations of HTML- and J2ME-prototypes of this application. The evaluations showed a positive acceptance of the approach and the prototype in general, although the results were strongly dependent on the used interaction technique. Touching NFC-tags was considered to be most reliable, innovative and easiest to handle, while taking photos of visual markers delivered the opposite results. The direct input of option values was seen as reliable and easy to use, but also boring and not very innovative.

3 A Framework for Mobile Service Interaction

The presented use-case scenarios and prototyping served as visualisations of a front end for mobile interaction with Semantic Web Services and as a first approach to the design of Physical Mobile Interactions. The posters respectively the physical objects and the mobile client application are part of a framework for mobile service interaction. Its purpose is to implement a coherent system that generates adapted interfaces from annotated Semantic Web Service descriptions in order to support mobile interaction with these services through Physical Mobile Interaction. The architecture of the framework is basically divided into 3 distinct parts (Fig. 3).

The *Physical Mobile Interaction Domain* of the framework includes mobile devices and client applications that interact with physical objects, for example posters. These objects are augmented with different technologies (e.g. NFC tags, visual markers or Bluetooth) that hold and provide information for the use and invocation of services that are again associated with the physical objects.

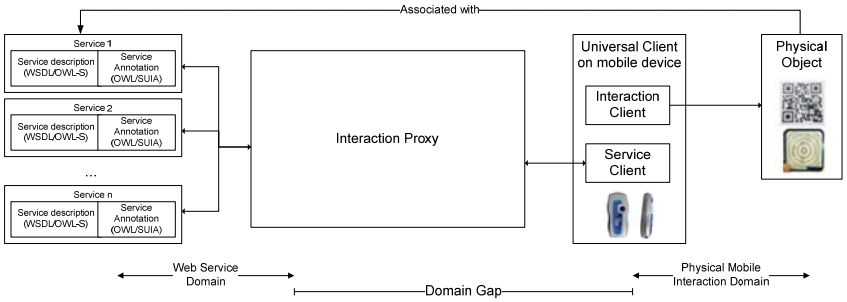


Fig. 3. Framework Architecture

These services can be found in the *Web Service Domain*. They are implemented as Semantic Web Services and represent the backend and service logic in the architecture. The descriptions of their functionalities, along with different extensions provide the basis for the generation of interfaces for Physical Mobile Interaction.

The *Interaction Proxy* mediates between Web Services and client applications. In order to increase the efficiency of the interface generation process, it is able to adopt and centralize common functionalities from the Semantic Web Services (e.g. service composition or reasoning) 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 open for the future integration and support of additional Web Services, client platforms and physical objects.

Depending on the target platform of the mobile device, 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 further rendering. The Universal Client itself works as a mediator between Semantic Web Services and physical objects. Its *Service Client*-component acts as an interface to the Interaction Proxy while the *Interaction Client*-component manages and abstracts different techniques for Physical Mobile Interaction with everyday objects. In addition, the Universal Client represents the generic application logic that renders and uses interfaces for this interaction and the invocation of associated Web Services.

The following chapters focus particularly on the annotation of Web Service descriptions in the Web Service Domain and their transformation into user interfaces using the Interaction Proxy.

4 Semantic Service Descriptions and Annotations

Our approach to service interaction requires services that can be easily described and extended with additional annotations to enable and support the automatic derivation of interfaces and to control their interaction flow. In order to meet these requirements

we take advantage of the flexibility and expressiveness of Semantic Web technology to model and describe services.

Fig. 4 gives an overview and a classification of the different service descriptions and extensions used within the framework. Although most descriptions are defined within the service domain, some of them are part of a shared knowledge between the components of the framework and hence independent from services, devices or physical interaction. They are extendable, reusable and essential in order to bridge the gap between Web Services and Physical Mobile Interaction as they describe common concepts that are related to each other and exist in both domains.

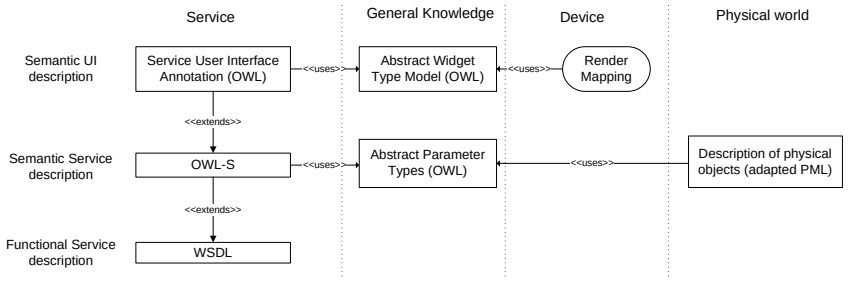


Fig. 4. Service descriptions and annotations used in the framework

The Web Services in the framework use the Web Services Description Language (WSDL) [4] to describe access to services and abstract message formats that are exchanged between them (e.g. names and types of input and output parameters). Since WSDL service descriptions are not suitable for deriving sufficient interface descriptions from them, the framework additionally exploits the OWL-S [5] service ontology as a basis for the user interface generation process. OWL-S builds upon WSDL and uses it as a concrete grounding for invoking services. An OWL-S description defines a set of one or more atomic processes which are the building blocks of the service process model. Atomic processes have inputs and outputs which are mapped to the corresponding WSDL description. They can also be further composed to composite processes which define specific control constructs such as sequence flows or parallel flows. Exploiting this OWL-S feature allows the modelling of different steps in more complex interactions. For example a sequential interaction flow of two steps can be modelled as two atomic processes composed into a sequence.

One of the main benefits of OWL-S is the definition and assignment of self-defined *Abstract Parameter Types* which are required for the correlation between physical objects and service parameters (see Fig. 4). Abstract Parameter Types are part of the general knowledge between physical objects and associated Web Services. This typing is crucial for relating information on a physical object to specific parameters of an associated service. Just as physical objects are associated with certain services (e.g. a movie poster for buying tickets via a ticketing service), information on these objects (e.g. options on a poster) are related to input parameters

of these services. This association is expressed through the common use of Abstract Parameter Types. The identification and description of information that is shared between objects and services is inspired by PML [6] (see Fig. 4), a markup language for specifying object properties that is used within the Internet of Things [3].

Since both WSDL and OWL-S are mostly abstract descriptions of how clients can invoke services, they don't provide enough information for the description and finally the rendering of user interfaces. Therefore the framework upgrades the standard OWL-S service descriptions with additional extensions that meet certain requirements:

- An abstract type system for interface widgets that facilitates their rendering and the graphical representation of service parameters.
- Concrete type formats and constraints for service parameters that have to be met by the information provided through Physical Mobile Interaction (e.g. date formats).
- Predefined sets of values that are valid for an input parameter type.
- Readable labels to increase the expressiveness and usability of interfaces
- Additional descriptions that explain the use and purpose of interfaces and widgets for the interaction with real world objects, e.g. getting a service parameter value by taking a picture of a visual marker.

The framework complements the standard OWL-S service descriptions with the OWL-S based *Service User Interface Annotation (SUIA)* ontology (see Fig. 5) which implements the postulated requirements for service description extensions.

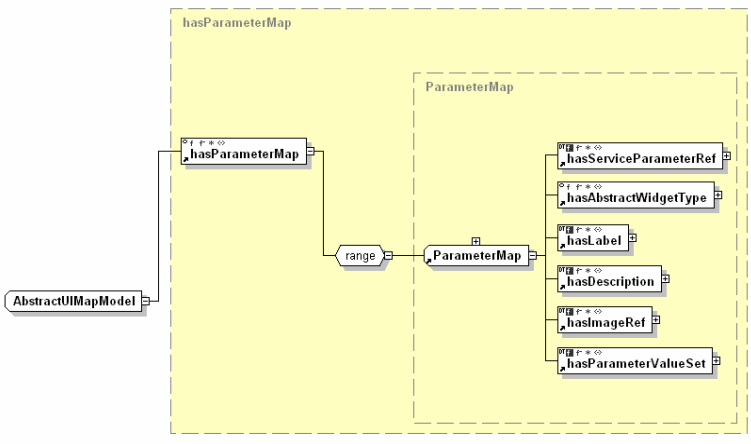


Fig. 5. Overview of the Service User Interface Annotation ontology (SUIA)

The main class of the SUIA for annotating an OWL-S service description is the *AbstractUIMapModel*. It serves as a collection bag for several parameter mappings and encapsulates all annotations for one service parameter. A single parameter mapping complements an input or output parameter in the OWL-S service description and indicates the required information for automatically generating a user interface.

Among these properties is an Abstract Widget Type that suggests the use of certain interface widgets for collecting specific parameters. It is interpreted by the interface rendering engine for the target platform and mapped to concrete user interface widgets. Fig. 6 shows the typology and the different widget types of the corresponding Abstract Widget Type Model that is specified in an external OWL model. It is also part of the general knowledge between the domains of Web Services and Physical Mobile Interaction and hence shared between them.

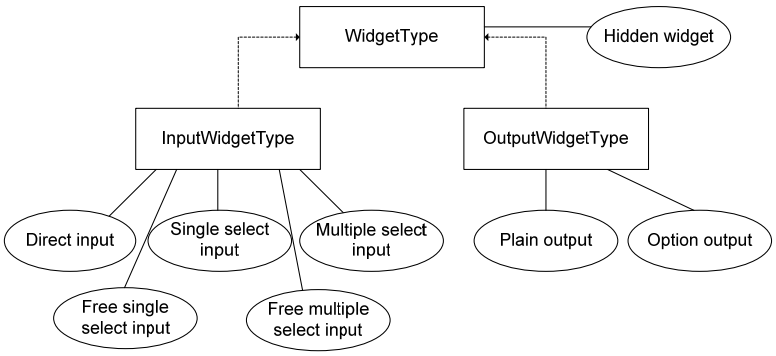


Fig. 6. The Abstract Widget Type Model

The hierarchy of 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. Instances of inputs are:

- *Direct input* widget, which provides an arbitrary input, e.g. via a text field.
- A *single select input* widget provides a single value from a given choice, such as a drop down or radio button menu.
- A *multiple selection input* widget provides several values from a given choice, e.g. from a checkbox.
- *Single selection* and *multiple selection* widgets allow a loose interpretation of the widget, meaning that a widget may be rendered as direct input or selection input.
- A *plain output* parameter widget will usually be rendered as a simple textual message.

- An *option output* contains a set of options for an interaction step. For example services may determine a set of options that was not predefined but calculated during runtime.
- A *hidden parameter type* is a special instance of the general class *WidgetType* and should not be rendered by the target platform.

5 User Interface Generation

Together with the basic OWL-S descriptions of Web Services, their SUIA extensions and the Abstract Parameter Type Model, the Abstract Widget Type Model provides the foundation and the input for the generation of interfaces for mobile service interaction. In order to be efficient, flexible and individual, this process has to balance 2 main requirements: The generation of interfaces from extended Web Service descriptions with the least effort and greatest reuse of resources as well as the generic support of different target platforms, rendering technologies, interaction techniques, user preferences and other context information. Especially these properties of the Physical Mobile Interaction Domain determine or restrain each other and have great influence on the generation of interfaces that invoke the same service but may look different from each other. The first prototype of the Interaction Proxy implements and supports the generation of interfaces for HTML- and J2ME-clients on mobile phones.

Fig. 7 gives an overview of the interface generation process within the Interaction Proxy. The first step is the composition of Web Service descriptions, extensions and type models into the *Abstract User Interface (UI) Description* that is the basis for the further generation of interfaces and their adaptation to client properties. Additional input may be provided by the service implementation itself as the output of a previous interaction step - e.g. a return value or a message on the outcome of a service method - may influence the next step.

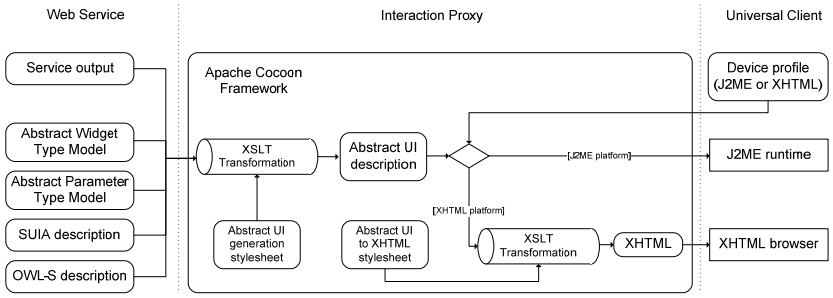


Fig. 7. Overview of the interface generation and transformation process

The Interaction Proxy gathers these descriptions and forwards them to Cocoon [7], a XML-framework for multichannel publishing that implements the generation of

interfaces and the transformation of their different descriptions. Cocoon aggregates these descriptions, applies the appropriate XSLT-stylesheet and derives an Abstract UI Description according to its transformation rules (see Fig. 7). This new, abstract description is basically a summary of the previous descriptions and contains all necessary information for the further interface generation while being more concise and easier to interpret (see Fig. 8).

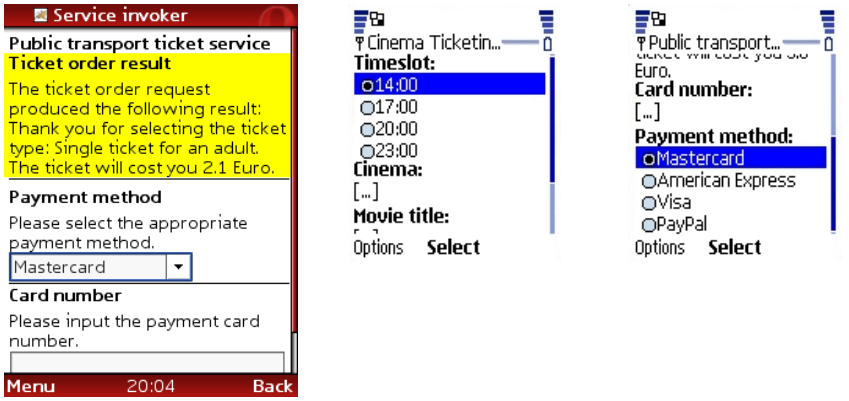
```
− <abstractUI>
  − <group title="Cinema Ticketing Service">
    − <widget type="http://perci.medien.ifi.lmu.de:8080/axis/ui/ParameterTypeModel.owl#singleSelectInputParameterType">
        <abstractType> http://perci.medien.ifi.lmu.de:8080/axis/domain/cinema/cinema.owl#Time </abstractType>
        <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.owl#directInputParameterType"></widget>
    </group>
  </abstractUI>
```

Fig. 8. Example for the Abstract User Interface Description

The next step is the rendering of a concrete user interface from the Abstract UI Description. This process can be implemented with either the Interaction Proxy or the Universal Client in the Physical Mobile Interaction Domain. The decision of where and how to render the interface is part of the Universal Client's interaction design that is determined by the supported platform, its technical resources, interaction techniques or user preferences.

The Interaction Proxy prototype implementation supports the generation of interfaces for clients that provide a mobile XHTML-browser or run J2ME midlets. In order to decide how to proceed in the interface generation process and where to render the interface, the Interaction Proxy takes context information into account that is passed with the first request for a specific Web Service interface. The current implementation of the Interaction Proxy recognizes information about the target platform from user agent headers (HTML browser) and Http-request properties (J2ME-clients) and has Cocoon use this information for choosing different branches of its multichannel publishing process (see Fig. 7). If the system recognizes a HTML user agent header, another transformation is applied to the Abstract UI Description. Another XSLT-stylesheet is used for a transformation in which the Abstract UI Description and its different widget-elements are translated into a HTML-document with input fields, checkboxes or drop-down-menus (see Fig. 9a). This document is returned to the Universal Client, interpreted and displayed by its HTML-browser. In

case the Universal Client is recognized to be supporting J2ME midlets, the Abstract UI Description itself is returned in order to be rendered by the J2ME runtime environment of the Universal Client application according to its own rules (see Fig. 9a). That way, the two different interfaces from Fig. 9 a/b allow interaction with the same Web Service, but are adapted to the properties of different client platforms.



a) **b)**

Fig. 9. Interfaces for HTML- (a) and J2ME-clients (b)

Both interfaces support Mobile Interaction to an extent that is determined by the technical resources of the corresponding client platform. Devices with HTML-browsers can only support the direct input of values from posters. J2ME-enabled mobile phones on the other hand can implement the recognition of visual markers or even provide interaction using Near Field Communication technology.

6 Related Work

We compared our approach to interface description, annotation and generation with different related work in order to get some inspiration and to define it as precisely as possible. For this purpose, we evaluated different user interface description languages (UIDL) and approaches to automatic interface generation, which often go hand in hand.

UIML [8] is a XML-compliant markup-language that was designed to describe and create user interfaces for applications on different devices and platforms. Its markup is independent of toolkits or programming languages but generic in order to describe interface elements in an abstract way. Different rendering engines implement the

generation of interfaces for specific target platforms and languages from UIML descriptions. Due to differences in the vocabularies of target languages, single UIML descriptions have to be tailored to the syntax supported by different rendering engines and can't be used to create multiple interfaces for different target languages [9]. Neither the generation of mobile user interfaces nor connections to Web Services are particularly supported.

The Personal Universal Controller (PUC) [10] that was developed within the Pebbles project [11] investigates how mobile devices like PDAs or cell phones can be used as remote controls for improving the interaction with common electrical appliances. For that purpose, mobile devices can download abstract specifications of their functionalities and use them for the automatic generation of user interfaces for controlling the corresponding appliances. This process also supports different interface modalities (graphical and speech), customisation to user preferences and the combination of controls for multiple connected appliances into a single interface.

[12] presents an approach that uses semantic descriptions from the OWL-S Service Profile and Process Model as a basis for the generation of dynamic form-based user interfaces. As some of their properties can not be derived from these descriptions, [12] extends Web Service ontologies with additional user interface annotations. Each OWL-S service description is associated with a UIModel extension that includes information about display labels, preferred widget types for providing the needed input or the grouping of fields and sub-fields.

While the idea behind Pebbles provides an interesting background for our approach, [12] confirms it and is the closest to its requirements as it exploits Semantic Web Service descriptions as a foundation for the generation of interfaces and utilizes user context information for this process.

7 Conclusion and Future Work

We presented a generic approach to improve mobile interaction with Web Services and a framework that integrates Semantic Web Service technology and Physical Mobile Interaction for this purpose and for their mutual benefit. Although the framework and the corresponding client application are still prototypes, this approach has generally been approved during two user studies. Future issues include extending the functionalities of the framework – the Interaction Proxy does e.g. not yet implement reasoning or service composition, modelling more information in the framework and its applications with Semantic Web technology and supporting more different client platforms.

8 Acknowledgements

The presented approach to the combination of Web Services and Physical Mobile Interaction was developed within the PERCI-project (PERvasive ServiCe Interaction) [13], a collaboration of the University of Munich and NTT DoCoMo Euro-Labs that is funded by the latter.

References

[1] Michael Rohs, Beat Gfeller. Using Camera-Equipped Mobile Phones for Interac ing with Real-World Objects. In: Alois Ferscha, Horst Hoertner, Gabriele Kotsis (Eds.): Advances in Pervasive Computing, Austrian Computer Society (OCG), Vi enna, Austria, 2004.

[2] Enrico Rukzio, Albrecht Schmidt, Heinrich Hussmann. Physical Posters as Gate ways to Context-aware Services for Mobile Devices. Sixth IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2004), English Lake Dis trict, UK, 2004.

[3] Steve Meloan. Toward a Global “Internet of Things”. 11.11.2003. 2003. http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/.

[4] Erik Christensen, Francisco Curbera, Greg Meredith, Sanjiva Weerawarana. Web Service Description Lanaguage (WSDL) Homepage. 2001. http://www.w3.org/TR/wsdl.

[5] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDer mott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, Katia Sycara. OWL-S: Semantic Markup for Web Services. 19.06.2006. http://www.daml.org/services/owl-s/1.1/overview/.

[6] PML The Physical Markup Language. 2002. http://web.mit.edu/mecheng/pml/index.htm

[7] The Apache Cocoon Project. Last published on 22.03.2006. http://cocoon.apache.org/.

[8] UIML Homepage. Last accessed on 03.07.2006. http://www.uiml.org/intro/index.htm.

[9] Marc Abrams, Constantinos Phanouriou. UIML: An XML Language for Building Device-Independent User Interfaces. XML '99, Philadelphia, 1999.

[10] Jeffrey Nichols. Automatically Generating User Interfaces for Appliances. In UIST 2004 Conference Companion, Santa Fe, NM, 2004.

[11] Brad Myers. Pebbles Homepage. 2005. http://www.pebbles.hcii.cmu.edu/.

[12] Deepali Khushraj, Ora Lassila: Ontological Approach to Generating Personalized User Interfaces for Web Services. International Semantic Web Conference 2005, 2005.

[13] http://www.hcilab.org/projects/perci/