International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:2, 2008

# Using the Semantic Web in ubiquitous and mobile computing: the Morfeo experience

José M. Cantera, Miguel Jiménez, Genoveva López, and Javier Soriano

***Abstract*—** With the advent of emerging personal computing paradigms such as ubiquitous and mobile computing, Web contents are becoming accessible from a wide range of mobile devices. Since these devices do not have the same rendering capabilities, Web contents need to be adapted for transparent access from a variety of client agents. Such content adaptation results in better rendering and faster delivery to the client device. Nevertheless, Web content adaptation sets new challenges for semantic markup. This paper presents an advanced components platform, called MorfeoSMC, enabling the development of mobility applications and services according to a channel model based on Services Oriented Architecture (SOA) principles. It then goes on to describe the potential for integration with the Semantic Web through a novel framework of external semantic annotation of mobile Web contents. The role of semantic annotation in this framework is to describe the contents of individual documents themselves, assuring the preservation of the semantics during the process of adapting content rendering, as well as to exploit these semantic annotations in a novel user profile-aware content adaptation process. Semantic Web content adaptation is a way of adding value to and facilitates repurposing of Web contents (enhanced browsing, Web Services location and access, etc).

***Keywords*—**Semantic Web, Ubiquitous and Mobile Computing, Web Content Transcoding. Semantic Markup, Mobile Computing Middleware and Services

## I. Introduction

THE development experienced over the last few years by mobile data communications has led to an increased demand for mobile applications and advanced services in the business world, especially in those business processes that involve people traveling. A big effort is also going into bringing the potential of this type of communications and applications on to the market for the general public, setting up ubiquitous and mobile computing as an emerging personal computing paradigm. The objective is to connect mobile people to the information and applications they need — anytime, anywhere, on demand; move the workplace to any space, using wireless middleware both on the server and on the client side to support the broadest spectrum of mobile networks and a wide array of devices on the client side; and to enable users of telephones and small, handheld, wireless computing devices to conduct business transactions or access information and services.

Despite the huge effort that is being invested in promoting such advanced services, uptake by users on a massive scale has not yet come about. In the coming years, the overall success of mobility technologies will largely depend on their level of penetration in the setting of professional and entrepreneurial applications and among the general public as a whole, as mobile computing products and information services delivering business value to the enterprise [1].

The development of mobility solutions in ubiquitous computing environments is conditioned by a variety of factors. These factors include the wireless nature of ubiquitous computing devices, device-inherent capacity restrictions, the range of features offered by the different mobile networks there are and the wide variety of available terminals and development technologies, with a range of configuration, functionality and rendering options, placing heavy demands on interoperability. Additionally, provision of mobile access to an application should ideally not lead to a modification of either the architecture or the business services published by this application. Hence, network architectures and software platforms are needed to support automatic, ad hoc configuration, service discovery and utilization by automated systems without human guidance or intervention, and capability description [2], as well as content adaptation for a wide range of mobile devices with different rendering capabilities [3], also called Web content transcoding [4].

Additionally, as pointed out by Ora Lassila in his keynote talk at IASW 2005 [5], the Semantic Web represents a means to improve the interoperability between systems, applications, and information sources. Emerging personal computing paradigms such as ubiquitous and mobile computing will benefit from better interoperability, as this is an enabler for a higher degree of automation of many tasks that would otherwise require the end-users' attention. Specific application areas of Semantic Web technologies with direct ramifications to these new paradigms include Web Services, context-awareness and policy modeling, as well as the inclusion of semantics (semantic markup) in the information rendered for users through the mobility platforms and in the actual Web content transcoding process.

Manuscript received February 25, 2006. This work is being supported in part by the CAM Education Council and the European Social Fund under their Research Personnel Training program, and by the Spanish Ministry of Industry, Tourism and Commerce under its National Program of Service Technologies for the Information Society (contract FIT-350110-2005-73).

José. M. Cantera is with Telefónica Research & Development, Spain (e-mail: jmcf@tid.es).

Genoveva López, Javier Soriano and Miguel Jiménez are with Department of Computer Science, Technical University of Madrid (UPM), Spain. (e-mail: {glopez, jsoriano}@fi.upm.es, mjimenez@pegaso.ls.fi.upm.es,).

Considering the advances achieved in the Semantic Web research area by the international research community, the strategy followed for building the existing mobile computing middleware solutions should be reconsidered and the possibility of including Semantic Web technologies and techniques should be examined, as should the benefits of such a decision. In particular, this paper addresses the role of semantic annotation in mobile computing software platforms in providing explicit semantics that can be understood by a content adaptation engine. This provides for both the semantic markup of the adapted Web content and the use of this semantics in the adaptation process to enhance the rendering delivered to the user according to his likes and dislikes, and therefore, optimize the information to be displayed on the device according to the user that the information targets. Although Web content annotation (or metadata) has a variety of potential applications [6], they can be categorized into three types: discovery, qualification, and adaptation of Web content. The primary focus of this paper is on Web content qualification and adaptation.

In this paper we describe Morfeo's Semantic Mobility Channel (MorfeoSMC), a reference architecture for applications incorporating mobility, which is being developed in the context of the Morfeo Open Source Software Community [7], led by Telefónica Research & Development, and which is concerned with the challenges described above. The availability of these components is a step forward towards bringing mobile data communications and advanced services to the public at large.

The remainder of the paper is organized as follows. Section II describes the Mobility Channel, an advanced components platform, which is the key component of the proposed architecture. This Mobility Channel will be able to be used to build low-cost mobility solutions in record time, with distinguishing features such as channel adapter-based multidevice access, the combination of pull and push communication paradigms, the possibility of automatically switching between online and offline operating modes, minimization of traffic and server connections. Then section III describes the Mobility Channel's potential for integration with the Semantic Web through the incorporation of semantic markup of the content in mobile applications developed based on this channel. This, it is stressed, enables a mobile device to be able to recognize and interpret the information contained in the application's renderings. It can therefore influence the navigation of the user handling the device and take actions on the information that the user receives, either by accessing other services, in applications within the same device, or as befits the nature of this information. Section IV focuses on the use of this semantics during the adaptation of the Web content for mobile devices. Section V gives a brief example of the use of MorfeoSMC to give mobile access to such a popular application as TPI's Yellow Pages. Finally, Section VI presents other related work.

## II. Mobility Channel Architecture

The Mobility Channel [8] (MC) is a vertical component-based platform for rapidly developing applications and services that can be used to create comprehensive and integrated mobility solutions while concealing the complexity involved in managing multiple devices.

This platform allows the development of applications and services according to a channel model supported by the principles of Service Oriented Architectures (SOA). This means that the services that implement the business logic and are run on the back-end should not be duplicated when another access channel (mobile devices) is added. A new software component, the channel adapter, has been conceived to achieve this goal. It is run on the front-end and conceals the peculiarities of a new more restricted means of access from developers and back-end services.

Mobile applications are written according to the "develop once, use many" paradigm. Therefore, renderings are defined in a single, XML-based language that can specify the visual controls making up each of the renderings, as well as the flow associated with the renderings, which contains the response to the different events fired by the visual controls.

The MC supports both navigation-based thin clients and Java code-based smart clients, communication being subject to http or WAP protocols in both cases. For smart clients, it also offers the possibility of sending *push* asynchronous notifications by sending messages to the mobile device. The rendering layer is organized around the Rendering Operations (RO), which define all the rendering flow and calls to Applications Operations (AO) needed to implement the application. The AOs are contact points with the application's business logic or back-end and are, therefore, the nexus between the MC and the application's business services.

Code-generating tools (markup, validation and behavior specification) are a key component of the MC. They generate, at development time, the pages that will cater for thin clients organized by the families and markup languages that the MC supports, i.e. XHTML-MP, WML, etc., or the code that the smart clients will execute. The generated ROs are customized for each specific device, taking into account each device's specific capabilities, such as the functions its navigator supports, screen size and organization or interface capabilities. The pages that cater for thin devices isolate the programmer from the tasks associated with multidevice programming: device recognition, paginating control, URL rewriting to maintain sessions, validation management, etc. This solution differs from other approaches to services provision for mobile devices that focus on transcoding the application HTML or XHTML pages at run time. Fig. 1 illustrates the Mobility Channel architecture and shows its main server-side components.

A key component of the Mobility Channel is the rendering definition language (RDL) [9]. RDL was developed to isolate developers from the details of the different mobile technologies rendering languages and serve as a basis for the

unified definition of renderings handled by the MC, whatever technology is ultimately used to display them. RDL is based on XML and is composed of a set of high-level visual controls, which will then be converted into rendering language items for each device family. Some of the visual controls are similar to HTML tags, like *head*, *body*, *title*, *hr*, etc. Others are also akin to HTML tags, but their behavior depends on the capabilities of the markup language to which they will be translated. Representative examples of these are *submit* and *action* for performing actions, *menu* and *select* for offering different options, *list* and *table* for displaying structured data, or *label*, and *textarea* for offering all the necessary text fields functionality. There are also controls aimed at organizing the rendering, like *panel*, *style* and *p*. For a full list and detailed explanation of visual controls, see [9]. The visual controls can be defined with all the data necessary for their representation, that is, can be self-contained, or can be defined to import certain data from the application context at rendering time. Sentences written in Expression Language (EL) embedded in rendering definition language attributes are used to import data from the context. EL is an access method to data stored in Java Beans included in the JSP 2.0 specification. Appearance is controlled by means of extended W-CSS [10] layout and style sheets, conserving attribute inheritance and overwriting.

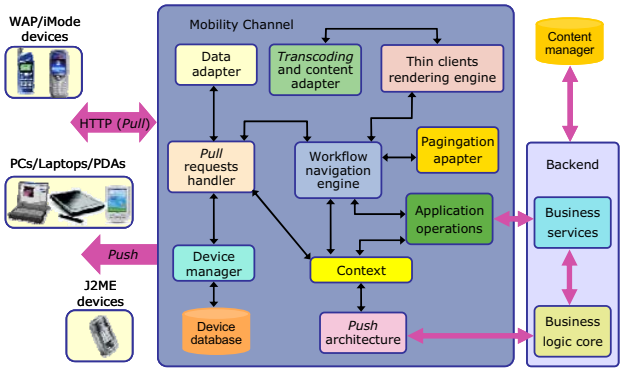


Fig. 1 Mobility Channel architecture

An example of visual control is given below. The control defines a table that will show the brand, model and price of vehicles that are in the object referenced by *${lcars}* present in the server *context*. The *context* (see Fig. 1) is a hierarchical data warehouse that contains the data handled by the mobile application at any one time. The visual controls always take their data from the context warehouse.

```
<table id="mytable" optionsbind="${lcars}" keymember="id">
   <th> <td>Brand</td>
        <td>Model</td>
        <td>Price</td> </th>
   <tr> <td member="brand"></td>
        <td member="model"></td>
        <td member="price"></td></tr>
</table>
```

The requests are first received by the *pull requests handler*, which is responsible for recognizing the device by calling up the *device manager* and storing the data from the mobile client (control used, event generated and event data) in the context. Then the *pull requests handler* has the *flow engine* start up the actions required to deal with the event triggered by the user in the mobile terminal. Generally, the actions will involve making a call to one or more *AOs,* which will leave new data in the *context*, to finally transform a navigation, in the case of thin clients, to a rendering served by the *rendering engine*. Before returning a page to the client, the contents are adapted to its capabilities and features, and these contents are paginated if necessary. In the case of smart clients, the process ends with a call to the *data adapter* that will serialize the data that the *smart mobile client* will receive.

The Mobility Channel is also equipped with an architecture for accessing offline operations, based on a microweb server that is hosted by mobile devices, enabling a later application-driven synchronization.

To be able to integrate data in a Content Management System into applications generated by the MC, the MC has been equipped with the capability of accessing content repositories compatible with JSR-170 (Java contents repositories access API). In this way, the same data as found in a CMS that is, for example, serving an enterprise's pages can be integrated into mobile applications or a Content Management System can be used to manage the data that will be displayed in the MC.

An example of a rendering in an RO is given below. The RO defines the "partner search" use case in a prototype enterprise mobile service. Fig. 2 shows three different screenshots, of three mobile devices with different sets of capabilities for this rendering.

```
<?xml version="1.0" encoding="UTF-8"?>
<cmt:document xmlns:cmt="http://morfeo-
project.org/tidmobile" id="query">
   <cmt:head>   <cmt:title>Buscar P/S</cmt:title>
          <cmt:style href="example1.css"></cmt:style>
   </cmt:head>
   <cmt:body style="corporativo">
      <cmt:p id="p1" layout="horizontal" cols="1">
         <cmt:link id="linkEmpresa">Empresa:</cmt:link>
         <cmt:label
style="remark">${enterprise}</cmt:label></cmt:p>
      <cmt:p id="p2" layout="grid" cols="2" style="nowrap">
         <cmt:label id="l1">Tipo P/S:</cmt:label>
         <cmt:select labelid="l1" style="req tiposps"
id="selTipo"
          bind="${sessionScope.searchData.typePS}"
optionsbind="${tiposPS}"

   beantype="org.morfeo.tidmobile.examples.bean.BeanSearchDa
ta">
         </cmt:select>
         <cmt:label>Nombre:</cmt:label>
         <cmt:entryfield id="enombre"
bind="${sessionScope.searchData.name}" style="nombre"
          beantype
="org.morfeo.tidmobile.examples.bean.BeanSearchData" />
         <cmt:label id="lf">Fecha:</cmt:label>
         <cmt:entryfield labelid="lf" id="efecha"
bind="${sessionScope.searchData.date}"
          style="fecha req"
beantype="org.morfeo.tidmobile.examples.bean.BeanSearchData"
          validationtype="java.util.Date" />   </cmt:p>
      <cmt:p id="p3" align="center"><cmt:submit
id="cconsulta" value="Consultar" /></cmt:p>
      <cmt:p id="p4" layout="horizontal">
         <cmt:link id="lvolver">Volver</cmt:link>
         <cmt:link id="linicio">Inicio</cmt:link></cmt:p>
   </cmt:body>
</cmt:document>
```

## III. Incorporation of Semantics into Mobile Applications

Just as a mobile device accessing an application or service receives data about the ROs that it will have to display to the user —text to be represented, concepts lists, page content

organization, etc.—, the client device can also receive semantic information about the data that the rendering contains. Therefore, the mobile application targets not only the user who reads those data, but the actual device is capable of recognizing information contained in the application renderings. Consequently, it can be party to the navigation of user the handling the device and undertake actions with the information that the user receives, either by accessing other services in applications within the actual device or as befits the nature of this information.

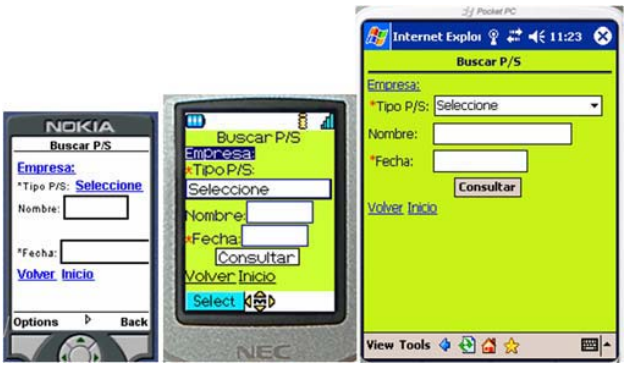


Fig. 2. Three screenshots for the same RDL

Like the actual data that the rendering will show to the user, there are two possible types of semantic information that we intend to attach to a rendering. On the one hand, the semantic information can be defined extensively during rendering, and therefore be statically associated with the rendering. In this way, when a delivery deadline is expressed, for example, it can also be expressed semantically that this delivery carries a given date as a value of the property that indicates the deadline. On the other hand, the semantic information to be sent to the client can come from structured data sources, in which case, like the information dynamically included in the renderings, they are referenced from the rendering using ELs, which reference variables and datasets. Therefore, the concepts displayed in tables, lists and other visual controls generated at run time will also be able to have associated semantic information.

Irrespective of how the semantic information is specified, two alternative mechanisms have been defined to describe the information in conjunction with the renderings definition.

1) The developer specifies semantic bindings associated with the visual controls that render information. To associate semantic information with data that are shown in the rendering, the rendering definition language for the *list*, *table*, *label*, *textarea*, *select* and *menu* visual controls has been extended to semantically describe the concepts that they display by means of additional attributes with which EL can be associated. There will also be implicit semantic information (e.g. dc:title). The semantic bindings will be expressed according to a set of imported ontologies. The mobility platform will be responsible for automatically generating RDF triplets associated with the bindings specified by the developer. Using the Mobility Channel tools, it will be possible to generate all the semantic information associated with an application. This mechanism's drawback is that any non-visualized knowledge will not be able to be marked semantically.
2) The developer specifies the semantics associated with the concepts in RDF/XML. It will be possible to specify semantic information about any type of information within a *metadata* tag at the end of a Mobility Channel rendering, even if this information is not part of the rendering and is not visualized in the device. This means that information present dispersedly in the display, in unstructured text, information related to what is displayed or additional information apart from what is displayed can be described semantically. Although, for the sake of simplicity or for reasons of space, this information should not be shown to the user, it is useful for the device to undertake other actions —see, for example, information about the latitude/longitude of a given establishment—. It will be possible to associate the EL with the semantic definition.

In either case, before returning an RO to the client, all the respective semantic information in the shape of RDF triplets will be included so that the client receives it together with the specific rendering data. Semantic annotations are sent to the mobile device within an external file to stop the content being mixed with metadata, as the W3C recommends. XPath [11] and XPointer [12] are used to associate annotating descriptions with annotated portions of a document [13].

### A. Semantic Bindings

The extension of the rendering definition language is the result of adding another five attributes, aimed at specifying the semantic information that will be generated together with the visual control with which they are associated:

- ***about-resid***: specifies what identifier one or more resources will take (by means of EL).
- ***about-class***: refers to the identifier of the RDF-S class to which a concept belongs.
- ***about-prop***: refers to the identifier of an RDF-S property, whose value is the datum represented in the visual control.
- ***about-obj-datatype***: defines the data type of an RDF literal, making use of XML-Schema types. It makes sense when the object is a literal.
- ***about-link-prop***: useful for the *table* visual control, it references the identifier of a property with a link to the class that contains the datum to be displayed in a column.

### B. Process of semantic annotation of visual controls

The following describes the process of semantic annotation of the different visual controls offered by the RDL. They all produce their own semantic annotation depending on their contents and their associated semantic bindings. In this paper, however, only the most representative examples and their semantic annotation will be explained.

The following examples use prefixes for the resources identifiers and for the imported ontologies that have been

declared using *import* sentences:

```
<cmt:import prefix="cars" uri=http://www.cars.net#
   file="cars-ontology.rdf" />
<cmt:import prefix="travel" uri=http://www.travels.net#
   file="travel-ontology.rdf" />
```

The language can also indicate a default prefix for generating resources identifiers as in:

```
<cmt:import prefix="myapp" uri=http://www.myapp.net
   default="true" />
```

*1) Table control*

The *about-class* attribute will be used to indicate the RDF-S class for mapping the items displayed in the table. The *about-resid* will be used to indicate how to generate the identifiers of resources associated with the table content, if there is no specification, they will be generated according to the *default URL + keymember* sequence. The table columns will have an "about-prop" attribute that will indicate the RDF property for mapping. If no *about-class* is specified at column level, the property will be understood to apply to resources of the type indicated by the *about-class* at table level. If a column contains information about a resource that is not of the type specified in *about-class* at table level, the *about-link-prop* attribute should also be specified at column level. This attribute indicates which property the table-type and column-type resources are linked to. As this specification, if necessary, will also generate resources identifiers, it will also be possible to generate an about-resid attribute at column level. If this attribute is not specified, blank RDF nodes may be generated.

Below we show the same example as proposed earlier, now including semantic information and the set of RDF triplets that will be generated.

```
<table id="mytable" optionsbind="${lcars}" keymember="id"
   about-class="cars:car"
   about-resid="myapp:cars:${lcars.id}">
   <th>
      <td about-prop="cars:brand">Brand</td>
      <td about-prop="cars:model">model</td>
      <td about-prop="cars:price"
         about-obj-datatype="xsd:float">Price</td>
      <td about-prop="terms:name" about-class="myorg:person"
        about-link-prop="terms:owner"
        about-resid="myapp:staff:${cars.owner.id}">Owner</td>
   </th>
   <tr>
      <td member="brand"/>
      <td member="model"/>
      <td member="price"/>
   </tr>
</table>
```

As the example shows, the notation can concatenate context data specified by means of EL with the URIs. (e.g. *about-resid=""myapp:cars:${lcars.id}*). The above specification automatically generates the following RDF triplets:

```
(myapp:cars/Id1, rdf:type, cars:car)
(myapp:cars/Id1, cars:brand, 'Seat')
(myapp:cars/Id1, cars:model, 'Leon')
(myapp:cars/Id1, cars:price,
   '22456'^^xsd:float)
(myapp:cars/Id1, terms:owner, myapp:staff:12772139D)
(myapp:staff/12772139D, rdf:type, myorg:person)
(myapp:staff/12772139D, terms:name, "Tim Berners-Lee")
(myapp:cars/Id1, xpath ont:reference, "...table[n]/tr[m]")
...[more items]...
```

Note that the last RDF triplet acts as a bridge between the semantic information about an item in the data table and a row in the markup table. This enables the visual elements that are described in the semantic information received together with the markup to be associated on the client side.

*2) List control*

The *about-class* attribute will be used to indicate which RDF-S class the displayed list belongs to. The *about-prop* attribute will be used to indicate which RDF-S property the information displayed in the list corresponds to. The *about-res-id* attribute (optional) will be used to indicate the identifiers of the RDF resources associated with the table content. If this attribute is not specified, default or blank RDF nodes identifiers will be generated. The *about-obj-datatype* attribute (optional) will be used to indicate the XSD type associated with the different objects of the property (predicate) represented by the list. An example of a list is shown below.

```
<list optionsbind="${lrestaurants}"
   about-class="travel:restaurant"
   about-resid="myapp:restaurants:${rests id}"
   about-prop="terms:name">
```

The list is also output from an element in the *context*, which is referred to as *${lrestaurants}*, and the respective RDF triplets are automatically generated from this specification:

```
(myapp:rests/Id1, rdf:type, travel:restaurant)
(myapp:rests/Id1, xpath ont:reference, "...ul[n]/li[m]")
(myapp:rests/Id2, rdf:type, travel:restaurant)
(myapp:rests/Id2, xpath ont:reference, "...ul[n]/li[o]")
...[more items]...
```

*3) Label control*

A label displays a datum. This datum will usually match a RDF literal (object). The *about-obj-datatype* attribute will type the RDF literal, if necessary. A specification of what property of what resource (or resource type) is being visualized is needed. The *about-class*, *about-resid* and *about-prop* attributes are the same as in the above elements. An example of a label is given below.

```
<label about-class="travel:hotel"
   about-resid="myapp:hotels:${hotel id}"
   about-prop="terms:name">Sheraton hotel</label>
```

*4) Select and Menu controls*

What the select/menu content represents should be indicated in both cases. The *about-class*, *about-prop*, *about-resid*, *about-obj-datatype* attributes are used for this purpose. An example of the semantic annotation of these controls is shown below.

```
<select id="s country" optionsbind="${countriesl}"
   keymember="iso" textmember="name"
   about-class="terms:country"
   about-res-id="myapp:country:${countriesl.iso}"
   about-prop="terms:name" />
<menu id="m payment" optionsbind="${means}"
   about-class="business:paymodes"
   about-res-id="myapp:payments:${means.kind}"
   about-prop="terms:name" />
```

*5) Image control*

The meaning of the image will be annotated externally by means of the *metadata* tag (databinding will be enabled). An example is given below.

```
<image>
     <metadata>
        RDF/XML block
     </metadata>
</image>
```

*6) TextArea control*

Two alternatives are provided for the semantic annotation of this control type. Annotation by means of an RDF/XML

*metadata* block associated with the textarea (databinding is enabled) and "inline" annotation of terms within the textarea

```
<textarea>Deadline of papers submission
   <aterm about-class="science:congress"
      about-resid="myapp:congress:ICOT2006"
      about-prop="science:deadlinedate"
      about-obj-datatype="xsd:date">25-01-2006</aterm>
</textarea>
```

Apart from the semantic information associated with the different controls of a Mobility Channel rendering, other RDF triplets will also be automatically generated containing general information, based, for example, on Dublin Core.

*7) Semantic digest*

The renderings will be able to incorporate semantic information about any type of information within a *metadata* tag at the end of a Mobility Channel rendering, even if this information is not part of the rendering and is not visualized in the device or is not specifically associated with the information present in any visual control. This means that information present dispersedly in the display, in unstructured text, information related to what is displayed or additional information apart from what is displayed can be described semantically. The *metadata* tag can be seen as a *bag* containing several kinds of semantic annotations, ranging from extensively defined semantic annotations in RDF/XML or OWL to elements that have some associated data binding, to generate the final annotations from data present in the *context*.

Using elements with some associated data binding is the method for defining semantic annotations to complement the annotations defining the visual elements that are present in *table*, *list*, *menu*, *select*, *image* and *textarea* items if we have specified the generation of an RDF identifier for their content, for example, all *table* rows. These annotations are written in a simplified *table* element inside the *metadata* block. This table contains no visual information for the user, but enables the definition of several properties (columns of this simplified *table*) of one or several items (*table* rows). This approach is compatible with the definition of the *table* visual control element and can be used to complement the semantic annotations of every visual control. However, no information is shown to the user because only semantic markup and no rendering markup is generated from the *metadata* tag.

```
<metadata>
   <rdf:RDF xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-
   syntax-ns#" xmlns:rdfs = "http://www.w3.org/2000/01/rdf-
   schema#" xmlns:dc = "http://purl.org/dc/elements/1.1/">
      <rdf:Description    about="http://www.myapp.net/pres1"
         dc:title="Cars" dc:description="Cars list"
   dc:publisher="Telefónica I+D"
      dc:date="2000-04-11" dc:language="en">
         <dc:creator><rdf:Bag>
              <rdf:li>Tim Berners-Lee</rdf:li>
              <rdf:li>Deborah L. McGuinness</rdf:li>
         </rdf:Bag></dc:creator>
      </rdf:Description>
   </rdf:RDF>
   <table id="RestSrvc" optionsbind="${lrestaurants}"
      about-class="travel:restaurant"
      about-resid="travel:restaurant:${lrestaurants.id}">
      <th>
         <td about-prop="trade:hasService"
            about-class="trade:service"
            about-link-prop="trade:offers" />
      </th>
   </table>
</metadata>
```

This semantic information produces the RDF triplets expressed in RDF/XML, as well as an OWL description of the trade services offered by the restaurants listed in the example given in section *3)*. Although it is not shown to the user, this information is very useful for rearranging the list items as explained in Section V. The following is the semantic markup generated in OWL language:

```
<owl:Class rdf:ID="RestSrvc2">
   <owl:equivalentClass>
      <owl:Class>
         <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
               <owl:onProperty>
                  <owl:ObjectProperty rdf:about="#offers"/>
               </owl:onProperty>
               <owl:someValuesFrom>
                  <owl:Class>
                     <owl:intersectionOf
                        rdf:parseType="Collection">
                        <owl:Class rdf:about="#RedWine"/>
                        <owl:Restriction>
                           <owl:onProperty>
                              <owl:ObjectProperty
                                 rdf:about="#hasRegion"/>
                           </owl:onProperty>
                           <owl:someValuesFrom
                              rdf:resource="#Rioja"/>
                        </owl:Restriction>
                     </owl:intersectionOf>
                  </owl:Class>
               </owl:someValuesFrom>
            </owl:Restriction>
            <owl:Restriction>
               <owl:onProperty>
                  <owl:ObjectProperty rdf:about="#offers"/>
               </owl:onProperty>
               <owl:someValuesFrom>
                  <owl:Class>
                     <owl:intersectionOf
                        rdf:parseType="Collection">
                        <owl:Class rdf:about="#Meat"/>
                        <owl:Restriction>
                           <owl:onProperty>
                              <owl:ObjectProperty
                                 rdf:ID="isCooked"/>
                           </owl:onProperty>
                           <owl:someValuesFrom
                              rdf:resource="#Fried"/>
                        </owl:Restriction>
                     </owl:intersectionOf>
                  </owl:Class>
               </owl:someValuesFrom>
            </owl:Restriction>
         </owl:intersectionOf>
      </owl:Class>
   </owl:equivalentClass>
</owl:Class>
...[more items]...
```

### C. Semantic Mobility Channel architecture

As a data source the MorfeoSMC will use a Content Management System with semantic capabilities. This will store structured contents as well as semantic information about these contents. Searches and queries of data not available in the MorfeoSMC context will be run on the content management to output semantic information that is included in the ROs sent to clients.

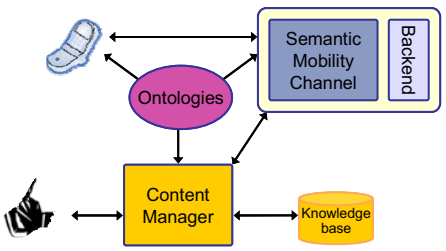


Fig. 3. Semantic Mobility Channel Architecture.

The content management system is also used to enter and edit contents, including semantic annotations of contents.

These are stored in conjunction with the data that they annotate. In this way, the knowledge base with which the content manager is associated contains all the application data, irrespective of whether they are structured content or semantic information. To enter and edit semantic contents, the content manager makes use of ontologies of the domains handled and thus customizes the interface, adapting it to the knowledge entered at any time.

These ontologies can likewise be used by mobile devices in conjunction with the information that they receive from the MorfeoSMC to make inferences on data.

## IV. Semantic Adaptation of Web Content

The adaptation of Web content explained so far targets the limited capabilities of mobile devices accessing Web applications or Web pages to assure that the final markup that is sent to the user is specially customized for his or her specific device. However, this is not the only possible content customization, The content can also be adapted bearing in mind the preferences of the user who is to read that information. In this case, the final rendering sent to the user is tailored with respect to the user's likes and dislikes.

To use a user's likes and dislikes in the content adaptation process, the user's preferences need to be modeled. This is done using *semantic user profiles* that express what concepts the user is interested in. Web contents that are intended to be adapted according to user preferences are the items that make up visual controls with extensive content, such as *tables* and *lists*, where the elements can be ordered and filtered depending on how well they relate to or match the concepts in the user preferences.

To give a clearer idea about how this matching works, this article focuses on trade services offered by enterprises and businesses and considers how well the user preferences are satisfied by the trade service, that is, the items of the table or list to be rearranged, prioritizing the businesses that offer trade services that better satisfy user interests.

The following semantic definitions and examples are expressed in description logic (DL) syntax [22] because it is more concise and relevant for expressing classes (concepts in DL) and properties (roles). Nevertheless, they have the same meaning and expressiveness as the more extensive definitions written in Semantic Web languages such as OWL [23].

For the semantic services and profiles to work together, the concepts involved in both elements must be described in the same ontology (i.e. vocabulary) or in mapped ontologies. A specimen ontology has been developed for the examples:

$$\begin{aligned} Thing &\sqsubseteq \top & Fish &\sqsubseteq Food \\ Wine &\sqsubseteq Thing & Region &\sqsubseteq Thing \\ RedWine &\sqsubseteq Wine & Rioja &\sqsubseteq Region \\ WhiteWhine &\sqsubseteq Wine & RiberaDuero &\sqsubseteq Region \\ Food &\sqsubseteq Wine & Cooking &\sqsubseteq Thing \\ Meat &\sqsubseteq Food & Fried &\sqsubseteq Cooking \end{aligned}$$

### A. Semantic user profiles

User profiles have long been used and are still being used to express user preferences and characteristics in data services. Every service provider used a customized user profile model that was more likely to its interests. Following this approach, a user used several heterogeneous profiles, one profile for every application or service he or she accessed, and every profile included specific data, which was similar to or very different from the data of the other the services identifying the same user. In this scenario, the concept of *semantic user profile* introduces a new framework in which the same profile can be used to access different services and applications. The semantic description of the concepts expressed in a user profile extends its use, which is no longer bound to the application for what it was originally designed.

A semantic user profile is a description of a user's long-term interests, as well as additional information such as identity, possessions, skills and knowledge or beliefs about one specific domain. For our purposes, we will focus solely on the first point, the user's long-term interests, namely, the user's likes and dislikes, which is the relevant information for prioritizing the elements that can best satisfy user preferences. Thus, a description logic expression of user interests is as follows:

$$\begin{aligned} Profile \equiv & \sqcap_i \exists hasInterest.Interest_i \\ & \sqcap_j \forall hasInterest.(\neg Disinterest_j) \end{aligned}$$

This definition is based on a single role or property called *hasInterest*, which is used to indicate the concept in which the user is interested, covered by the $Interest_i$ group, and the concepts in which the user is not interested, expressed by $Disinterest_j$. This definition is based on the definitions proposed by [24],[25].

The following represents a sample profile of a user stating that he likes fried food and *Rioja* wines, but is not interested in *Ribera del Duero* wines.

$$\begin{aligned} User1 \equiv & \exists hasInterest.(Food \sqcap (\exists isCooked.Fried)) \sqcap \\ & \exists hasInterest.(Wine \sqcap (\exists hasRegion.Rioja)) \sqcap \\ & \forall hasInterest.(\neg Wine \sqcup \neg(\exists hasRegion.RiberaDuero)) \end{aligned}$$

One of the key features of this approach is that the concepts used in the user profile can be refined as much as you like to express the inherent complexity of the user preferences.

### B. Semantic trade services

The user preferences can be compared and arranged in relation to the semantic descriptions of items in *lists* and *tables*. This comparison (or semantic matching) can be made against the semantic descriptions of the Web content generally. However, this article focuses on a specific content description, namely, the trade services provided by companies or enterprises, which are the goods or products that the enterprise offers to its potential clients. Companies can also offer services (in their broadest sense), which can be modeled as concepts in the same way as products can.

The meaning of the word *service* has need of further

clarification. This section refers to the semantic description of trade services, which is not the same thing as the semantic descriptions of Web Services that are dealt with by OWL-S or WSMO and have a different goal.

Trade services are concepts of the following form:

$$Service \equiv \exists offers.Thing$$

This definition confines a trade service to being anything that provides *Things*, and also serves as a definition for the service concept. This simple expression does not explicitly specify the definition of the special characteristics of the services, which can be defined as features of the offered Things. The following example makes this point.

$$RestSrvc1 \equiv \exists offers.(WhiteWine \sqcap (\exists hasRegion.Rioja)) \sqcap \exists offers.Meat$$

$$RestSrvc2 \equiv \exists offers.(RedWine \sqcap (\exists hasRegion.Rioja)) \sqcap \exists offers.(Meat \sqcap (\exists isCooked.Fried))$$

$$RestSrvc3 \equiv \exists offers.(RedWine \sqcap (\exists hasRegion.RiberaDuero)) \sqcap \exists offers.Fish$$

This example shows a simplification of the trade services offered by several restaurants, which is, however, expressive enough to convey the potential of the description.

This information can be gathered during the content adaptation process, when the semantic annotations related to the visual controls are generated, as explained in Section III.

### C. Semantic matching

The semantic definition of the user profile or the semantically enriched descriptions of the Web contents are of no use without a procedure that can determine whether the Web contents fit the profile. This is done by *semantic matching* between the concepts expressed in the user profile and the Web contents. In this paper we present a *semantic matching* between user preferences and trade services to determine whether the user profile is semantically compatible with a particular trade service and, if so, how well the two match. This information is very useful for adapting the contents of *lists* and *tables*, enabling the generation of the best arrangement of companies tailored for a particular user.

In terms of ontologies, the concepts describing a user's likes and dislikes have to be compared to the concepts supplied by a trade service:

$$UserInterest \equiv \exists hasInterest^{-}.Profile$$

$$ServiceOffer \equiv \exists offers^{-}.Service$$

The match between *UserInterest* and *ServiceOffer* is a measure of how well the services supplied satisfy the user's preferences, and needs to be calculated for every <user, offered service> pair. To determine the degree of match we use the levels proposed in [26], which are listed from best to worst:

- ***Exact*** if *UserInterest* is equivalent to *ServiceOffer*.
- ***PlugIn*** if *UserInterest* is sub-concept of *ServiceOffer*.
- ***Subsume*** if *UserInterest* is super-concept of *ServiceOffer*.
- ***Intersection*** if the intersection between *UserInterest* and *ServiceOffer* is satisfiable.
- ***Disjoint*** if *UserInterest* and *ServiceOffer* are completely incompatible.

The elements inside a *list* or *table* can be rearranged using the results of the matching process to make the most relevant items appear in the top positions or pages.

According to the proposed semantic matching, the examples of restaurants shown above would be rearranged as follows: {RestSrvc2, RestSrvc1, RestSrvc3}.

## V. Application example

The MorfeoSMC is being used to give such a widespread application as TPI's Yellow Pages [21] mobility, as well as aiming to offer search results with semantic information based on location, services and products ontologies. The objective is twofold. On the one hand, the aim is to extend service accessibility to the whole range of mobile devices, which is in line with the philosophy of any yellow pages service, where mobility services and activities search can account for a large proportion of accesses to this application. On the other hand, it aims to take this service further, by sending the results with associated semantic information. In this way, the user is not the only consumer of this information, but his or her device is able to interpret it, integrate this knowledge with other applications accessible in the device —thereby maximizing the potential for TPI use—, or be proactive and suggest what different actions should be taken in relation to the concepts handled in navigation. On this point, it is worth highlighting the potential offered by Semantic Web Services in conjunction with the semantic information obtained from navigation, integrating access to the SWS in actual navigation.

## VI. Related work

As regards software platforms, a range of advanced development platforms have been proposed in response to some of the challenges raised by mobile environments. Proposals like IBM's WebSphere EveryPlace Access [14] based on WebSphere Transcoding Publisher technology [15] and on ideas developed in [16] on visual XPath expressions for external metadata authoring composition, MobileAware's Mobile Interaction Server and Mobile Content Proxy products [17] or Microsoft's MS Mobile Controls [18] are leading examples. These platform technologies will improve the design of advanced mobile applications that are tailored for every mobile device. To our knowledge, however, none of them include mechanisms for incorporating semantics in the generated renderings, which is a key element for their integration with the Semantic Web and the exploitation of all its potential.

Although overlooked by existing mobility platforms, several approaches have been developed that consider the possibility of including Semantic Web technologies and techniques for building the next generation of semantic-aware mobile computing middleware solutions. For example, [19] presents a discussion of how Semantic Web technologies can

be used in the context of ubiquitous computing to enrich the capabilities of service discovery mechanisms and to enable the device coalitions that opportunistically exploit a dynamically changing ubiquitous computing environment.

Along the lines of this paper, [3] introduces a RDF-based framework of external annotation applicable to transcoding for Web-enabled mobile devices and explains a high-level overview of RDF annotation-based transcoding. The proposal contemplates the idea of authoring-time transcoding. The work has grown out of two previous papers [19] and [13], and it focuses on the use of RDF in supporting the transcoding process. However, the role of annotation in this work is to characterize ways of adapting content rather than to describe the contents of individual documents themselves.

[20] shows how when HTML documents are translated into multiple target languages by means of a machine translation engine (i.e. a transcoder), linguistic annotations would be helpful for improving the translation accuracy. Once again, the approach does not address the semantic annotation of the content.

Unlike the above work, all based on run-time transcoding mechanisms, the Mobility Channel tools generate the pages that will cater for requests from each family of mobile devices, i.e. XHTML-MP, WML, etc., at development time. Additionally, none of the above papers addresses the generation of semantics associated with the information that is sent to mobile clients in different ROs, which is the primary original contribution of this paper.

## REFERENCES

[1] S. Chughtai and L. M. Patterson, "Robust mobile-computing products delivering business value to your enterprise." Whitepaper G224-9130-00, *IBM Pervasive Computing*, October 2004.
[2] O. Lassila and M. Adler, "Semantic Gadgets:Ubiquitous Computing Meets the Semantic Web," In D. Fensel, et al. Eds. *Spinning the Semantic Web*, The MIT Press, Cambridge, Ma., pp. 363-376, 2003.
[3] M. Hori, G. Kondoh, K. Ono, S. Hirose, and S. Singhal, "Annotation-based Web Content Transcoding", *Proc of the 9th Int World Wide Web Conference (WWW9)*, Amsterdam, 2000. Available: http://www9.org/.
[4] K. H. Britton, et al., "Transcoding: Extending E-Business to New Environments," *IBM Systems Journal*, 40(1), pp. 153-178, 2001.
[5] O. Lassila, "Using the Semantic Web in Ubiquitous and Mobile Computing," Keynote, IASW05, Jyväskylä, Finland, 25-27 Aug. 2005.
[6] O. Lassila, "Web Metadata: A Matter of Semantics," *IEEE Internet Computing*, no. 2, vol. 4, pp. 30-37, 1998.
[7] Morfeo Project: Open Source Community for Software Platforms and Services Development. http://www.morfeo-project.org.
[8] J. Cantera "Enterprise Mobility Solutions: Technologies, Components and Applications," *Communications of TID*, 36, June 2005.
[9] TIDMobile: Presentation Definition Language Reference Guide (Revision 1.2.1). Available: http://www.morfeo-project.org/files/ /TIDMobile_LanguageReference.pdf, September 2005.
[10] Open Mobile Alliance, WAP CSS Specification, WAP-239-WCSS-20011026-a, 2001.
[11] World Wide Web Consortium, XML Path Language (XPath), 1999. Available: http://www.w3.org/TR/xpath.
[12] Workd Wide Web Consortium. XML Pointer Language (XPointer), 2002. Available: http://www.w3.org/TR/xptr.
[13] M. Hori, "Semantic Annotation for Web content Adaptation. In D. Fensel, et al. Eds. *Spinning the Semantic Web*, The MIT Press, Cambridge, Massachusetts, pp. 403-429, 2003.
[14] WebSphere EveryPlace Access. Available: http://www-306.ibm.com/ /software/pervasive/ws_everyplace_access/.
[15] IBM Corporation, WebSphere Transcoding Publisher, 2001. Available from http://www.ibm.com/software/webservers/transcoding/.
[16] M. Abe and M. Hori, Visual Composition of XPath Expressions for External Metadata Authoring," RT-0406, IBM Research, Tokyo, 2001.
[17] MobileAware Mobile Interaction Server and Mobile Content Proxy. Available: http://www.mobileaware.com.
[18] Microsoft Mobile controls. Available: http://msdn.microsoft.com/ /mobility/othertech/asp.netmc/default.aspx
[19] M. Hori, K.Ono, G. Kondoh, and S. Singhal, "Authoring Tool for Web Content Transcoding," *Markup Languages: Theory and Practice*, 2(1), pp. 81-106, 2000.
[20] K. Nagao, Y. Shirai, and S. Kevin, "Semantic Annotation and Transcoding: Making Web Content More Accessible," *IEEE* Multimedia, no. 8 vol. 2, pp. 69-81, 2001.
[21] TPI Yellow Pages service, Telefónica. Available: http://www.tpi.es/
[22] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P.F. Patel-Schneider eds. "The Description Logic handbook: Theory, Implementation and Applications." Cambridge University Press, 2003.
[23] F. Baader, I. Horrocks, and U. Sattler. "Description logics as ontology languages for the semantic web." *D. Hutter and W. Stephan, editors, Festschrift in honor of J org Siekmann, Lecture Notes in Artificial Intelligence*. Springer, 2003.
[24] A. Calì, D. Calvanese, S. Colucci, T. Di Noia, and F. M. Donini. "A description logia based approach for matching user profiles." *Proceedings of the 2004 Description Logic Workshop (DL 2004)*, 2004.
[25] A. Hessling, T. Kleemann, and A. Sinner, "Semantic User Profiles and their Applications in a Mobile Environment." *AIMS workshop Ubicomp Conference 2004*, Nottingham, England, September 2004.
[26] L. Li and I. Horrocks. "A software framework for matchmaking based on semantic web technology". Proceedings of the 12th International Conference on the World Wide Web, Budapest, Hungary, May 2003.