World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:1, No:11, 2007

Semantic Mobility Channel (SMC): Ubiquitous and mobile computing meets the Semantic Web

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 is exploited for either an individual element or a set of consecutive elements in a Web document and 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 SMC, enabling the development of mobility applications and services according to a channel model based on the principles of Services Oriented Architecture (SOA). It then goes on to describe the potential for integration with the Semantic Web through a novel framework of external semantic annotation that prescribes a scheme for representing semantic markup files and a way of associating Web documents with these external annotations. 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. Semantic Web content adaptation is a way of adding value to Web contents 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, including 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

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,).

users through the mobility platforms and in the actual Web content transcoding process.

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 order to provide explicit semantics that can be understood by a content adaptation engine, enabling the semantic markup of the adapted Web content. 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 the Web content qualification and adaptation.

In this paper we describe a Reference Architecture for applications incorporating mobility, which is being developed in the context of the Morfeo Opensource 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, and which 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, and 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 briefly exemplifies the use of SMC to give such a widespread application as TPI's Yellow Pages mobility. Finally, Section V 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 (mark-up, 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 mark-up 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. The Fig. 1 illustrates the Mobility Channel architecture and shows its key 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*, *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.

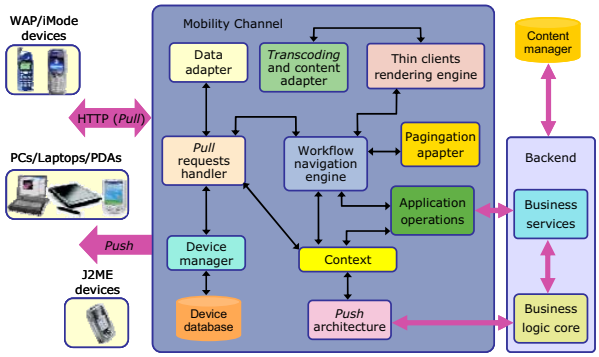

Fig. 1 Mobility Channel architecture

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 micro web 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.

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, and it can be party to 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.

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 be therefore 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 data sets. 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 it 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 or associated with any visual control, 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 this information should not be shown to the user for reasons of simplicity or shortage of space, it is useful for the device to undertake other actions —see for example information about the length/width of a given establishment—. It will be possible to associate EL with the semantic definition.

In either case, before returning a 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, in order to avoid the mixture of content and 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 RDL.

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>
        <td member="model"></td>
        <td member="price"></td></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:

```
(http://www.myapp.net/cars/Id1,rdf:type,cars:car)
(http://www.myapp.net/cars/Id1,cars:brand,'Seat')
(http://www.myapp.net/cars/Id1,cars:model,'Leon')
(http://www.myapp.net/cars/Id1,cars:price,
  '22456'^^xsd:float)
(http://www.myapp.net/cars/Id1,terms:owner,myapp:staff:12772
139D)
(http://www.myapp.net/staff/12772139D,rdf:type,myorg:person)
(http://www.myapp.net/staff/12772139D,terms:name,
  "Tim Berners-Lee")
(http://www.myapp.net/cars/Id1,xpath ont:reference,"...table
[n]/tr[m]")
```

The example uses 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" />
```

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" />
```

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 about-class="travel:hotel" about-prop="terms:name">
```

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. *about-class*, *about-resid* and *about-prop* attributes as in the above elements. An example of label is given below.

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

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 associated "summary" semantic information. This information will be able to be automatically downloaded (without the need for user intervention) by a navigator when there is a link from another rendering pointing to this information. This information will be able to be defined by means of a *metadata* block associated with the rendering definition *head* block.

```
<cmt:head>
   <cmt:title style="noinclude">Validation</cmt:title>
   <cmt:style href="example1.css"></cmt:style>
   <metadata>RDF/XML Block</metadata>
</cmt:head>
```

8) Semantic information

The Mobility Channel renderings will be able to incorporate a *metadata* tag containing semantic information in RDF/XML format at the end. Data binding will be applicable to such tags.

```
<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>
</metadata>
```

C. Semantic Mobility Channel architecture

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

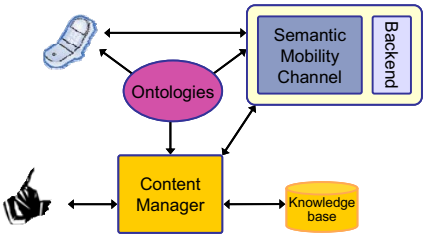

Fig. 2. Semantic Mobility Channel Architecture.

The content management 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 SMC to make inferences on data.

IV. Application Example

The SMC 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 different actions in relation to the concepts handled in navigation be taken. 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.

V. 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/