The Web Mashup Scripting Language Profile

Marwan Sabbouh , Jeff Higginson, Caleb Wan, Salim Semy, Danny Gagne

1.0 Introduction

The Web Mashup Scripting Language (WMSL) [1] enables an end-user ("you") working from his browser, e.g. not needing any other infrastructure, to quickly write mashups that integrate any two, or more, web services on the Web. The end-user accomplishes this by writing a web page that combines HTML, metadata in the form of mapping relations, and small piece of code, or script. The mapping relations enable not only the discovery and retrieval of the WMSL pages, but also affect a new programming paradigm that abstracts many programming complexities from the script writer. Furthermore, the WMSL Web pages or scripts those disparate end-users ("you") write can be harvested by crawlers to automatically generate the concepts needed to build aligned ontologies. These aligned ontologies are comprised of the local semantics of web services' data models, extentions of context ontologies (middle ontologies), and links, or mappings, between the data models and also to the middle ontologies.

In general the WMSL script contains four blocks:

1. Imports of Web Service Description Language (WSDL) files [2], schemas, ontologies, and other WMSL scripts;
2. Alignments of entities and concepts;
3. Workflow statements; and
4. Mediation statements that can possibly be followed by other workflow statements.

Each of the major blocks constituting the WMSL webpage or script can be encoded either in HTML or scripting. For the purpose of this paper, we discuss the WMSL-Profile; or the encoding of the import block, and the alignments of entities and concepts block in HTML of a WMSL web page. That is, we describe the conventions of encoding and of parsing the WMSL-Profile. We also describe the automatic generation of aligned ontologies from the WMSL-Profile. It is envisioned that WMSL webpages are created by end-users, and the parsing of the WMSL webpages, which yields the aligned ontologies, is accomplished by a user agent.

Figure 1 shows a WMSL Web page for a use case that was presented in [3] and [4]. The use case discusses the integration of two air flight systems: Air Mobility (AM), and Air Operations (AO). The AM system is responsible for many different types of missions including: mid-air refueling, the movement of vehicles, and the tasking of Air Force One. The AO system is primarily concerned with offensive and defensive missions. Each system was developed independently and built for the specific needs of the users. In both systems, a mission is represented as a set of position reports for a particular aircraft.

```
<html>
        <head profile="http://mitre.org/wmsl/profile">
                <title>WSML Use Case</title>
                <base href=" http://mitre.org/owl/1.1/"/>
                <link rel="schema.AM" type="text/xml"
                        href="http://www.mitre.org/xsd/1.1/AM#"/>
                <link rel="schema.AO" type="text/xml"
                        href="http://www.mitre.org/xsd/1.1/AO#"/>
        </head>
        <body>
                <dl class="owl-equivalentClass">
                        <dt><a href="AM#CallSign">AM#CallSign</a></dt>
                        <dd><a href="AO#CallSignName">AO#CallSignName</a></dd>
                </dl>
                <dl class="owl-sameAs">
                        <dt><a href="AM#A10A">AM#A10A</a></dt>
                        <dd><a href="AO#A010A">AO#A010A</a></dd>
                </dl>
                <dl class="mappings-match">
                        <dt><a AM#AircraftType">AM#AircraftType</a></dt>
                        <dd><a AO#AircraftType">AO#AircraftType</a></dd>
                </dl>
                <dl class="mappings-hasContext">
                        <dt><a href="AO#AOCoord">AO#AOCoord</a></dt>
                        <dd><a href="position#Coord-GEODETIC-WGE">
                                position#Coord-GEODETIC-WGE</a></dd>
                </dl>
                <dl class="mappings-hasRelation">
                        <dt><a href="position#Coord-UTM-WGE"></a></dt>
                        <dd><a href="position#UTM"></a></dd>
                </dl>
                <dl class="rdfs-subclassOf">
                        <dt><a href="AM#A10A"></a></dt>
                        <dd><a href="AM#AircraftType"></a></dd>
                </dl>
        </body>
</html>
```

Figure 1 A Sample WMSL Profile for the AM-AO Use Case

In this scenario these two systems will be integrated so that the AO system can be kept apprised of all the AM missions. We write a WMSL to accomplish this integration, focusing on the first two blocks of the

WMSL. The WMSL yields the aligned ontologies of the AM and AO necessary to reconcile syntactic, structural, and representational mismatches between the AM and the AO schemas which was demonstrated in [3] and [4]. It is important to note that the import block of the WMSL-Profile, which also yields the ontological description of web services necessary for their automatic invocation and for handling their response, will be addressed in a future paper. First the WMSL imports the WSDL files of the AM and AO, and the WSDL of shared context which is the GeoTrans translator service that translates between geo-coordinate systems. Then, the WMSL uses six mapping relations to align entities between the AM and AO schemas and for their mappings to the WSDL of Geotrans [5]. Not coincidentally, these are the same mapping relations that we have used in our previous work [3] and [4], with the main difference being now they are specified in the WMSL web page rather than in the ontologies. These mapping relations define three mapping patterns that are used to reconcile syntactic, structural, and representational mismatches between data models. The mapping patterns which are discussed in greater detail in our previous work and are shown in Figure 2. The mapping relations are:

- owl::equivalentClass
- owl::sameAs
- rdfs::subclassOf
- hasMatch
- hasContext
- hasRelation

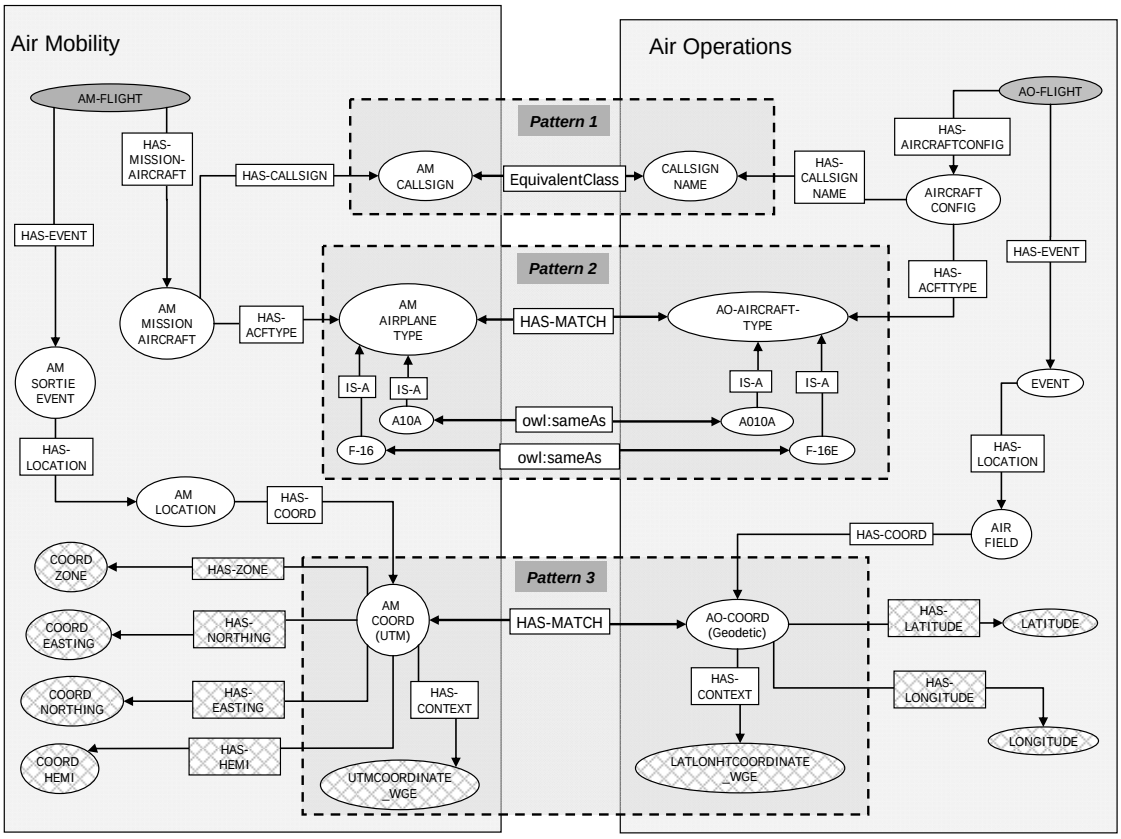

Figure 2 Aligned Ontologies for the AM-AO Use Case

The first three relations are used in accordance with the specifications that they were taken from. The hasMatch, and hasContext relations are needed in order to resolve structural, syntactic, and representational mismatches between the legacy schemas. The *hasRelation* establishes a generic relationship between a subject and an object. The import of the WSDL and the existence of these mapping relations in the WMSL enables an open-source model of building aligned ontologies. In this paper, we discuss the WMSL-Profile, and the generation of the aligned ontologies. The remainder of the WMSL, which deals with specifying the order of invocation of the web services, and of invoking the mediator, will be addressed in a future paper.

This paper proceeds as follows: in the next section, we describe the purpose and motivation behind the creation of this technology. In section 3, we describe the encoding conventions of WMSL followed by the parsing conventions of WMSL using the AM and AO example in section 4. In section 5, we relate this approach to the literature, discuss its implications, and conclude the paper.

2.0 Motivation

On the web, we are faced with increased challenges in adopting semantics. Technologies such as OWL [6] and RDF [7] have not enjoyed the wide adoption that was once anticipated. Service oriented architectures (SOA) are losing to 'light" approaches based on Microformats [8], Ajax [9], RSS [10], and REST [11]. There are several reasons for this state of affairs. In particular, the above-mentioned lightweight technologies are fairly simple to use. The benefits of these technologies include the creation of mashups using Ajax, the specifications of lightweight semantics with Microformats, and the timely notification of new content using RSS. More importantly, these new technologies have enjoyed the backing of millions of Web developers, and have the momentum going forward.

On the other hand, the adoptions of Semantic Web technologies have been slowed by the lack of a killer application that has Web scale. To the best of our knowledge, no one has demonstrated a Semantic Web application that has Web-scale. In other words, although many advantages have been articulated for the Semantic Web, these advantages can also be achieved using other, and often simpler, technologies. Furthermore, upon closer inspection it is becoming clearer that light semantics are sufficient in many applications. For example, in our previous work we concluded that we can achieve workflow automation from the pair-wise mappings of data models and from their mapping to some shared context, regardless of whether OWL\RDF, XML Schemas, or UML is used to describe the data models. Adding to these issues are the multiple competing standards in Semantic Web Services [12] such as OWL-S [13], WMSO [14], and others, and how they can be harmonized with existing W3C standards. Another issue is the lack of social processes for the design of ontologies as is the case for Folksonomies or in the social tagging case.

Yet, we cannot escape the fact that semantics are absolutely needed to enable automated reasoning on the web, or to enable information exchange in the enterprise. Even today there is no consensus on how to best specify semantics. Those who are in the XML schema camp suggest that a well-designed XML schema contains rich semantics. They point to the fact that they don't need to redesign their schemas in OWL and RDF. Others are adamant that semantics must be captured formally using OWL and RDF, and that the informal semantics captured by XML schema is not sufficient. We believe WMSL, when combined with existing schemas such as WSDL files, offers sufficient semantics for many applications. That is, WMSL leverages all the semantics that exist in XML schemas while offering the facilities to assert further semantics that may be missing from XML Schemas. This positions WMSL as the glue that takes in XML schemas and yields formal ontologies. Since WMSL is HTML and scripting, it therefore has Web scale. Furthermore, WMSL is lightweight as it can be run from a browser. Finally, our solution enables a light SOA approach where anyone can write a WMSL script to implement a mashup in support of information sharing requirements. Furthermore, WMSL can automatically generate the semantics needed to index and search structured data as is done with free text today.

3.0 WMSL-Profile Specifications

3.1 Encoding of the Schema Declarations Using the Header Block

In this section, we specify the conventions used in encoding the WMSL-Profile. Before we proceed, we would like to point out that other encodings are certainly possible, and the one presented here may not be the best. However, it is important to recognize that WMSL uses the standard HTML tags, and does not introduce new HTML tags.

To direct the user agent to follow the WMSL conventions in parsing this document, we use the standard HTML profile attribute of the head tag as shown in Figure 3.

```
<head profile=http://mitre.org/wmsl/profile>
```

Figure 3 Use of the Profile Attribute

To declare the WSDL files employed by the integration, we use a method compliant with that used by the embedded RDF specification as well as the method used by the Dublin Core to embed metadata in HTML using the *link* and *meta* tags. Specifically, we use the *rel*, *type* and *href* attributes of the *link* tag. The general pattern is shown in Figure 4 followed by examples in Figures 5 and 6:

```
<link rel="schema.prefix" type="MIME Content Type" href="uri" />
<link rel="schema.AM" type="text/xml" href="http://www.mitre.org/xsd/1.1/AM#"/>
<link rel="schema.AO" type="text/xml" href="http://www.mitre.org/xsd/1.1/AO#"/>
```

Figure 4 Import of the WSDL files using the Link tag

The above statements also declare a schema prefix that can be used later in the HTML. Next we declare the schema of the mapping relations. It is important to note that we set the type value to "application/rdf+xml" to denote that it is an ontology file.

```
<link rel="schema.map" type="application/rdf+xml"
       href="http://www.mitre.org/mappings/1.1/mappings#"/>
```

Figure 5 Use of the type Attribute

Next, we need to specify the handle for using relations from the RDFS and OWL specifications:

```
<link rel="schema.owl" type="text/html" href="http://www.w3.org/2002/07/owl#"/>
<link rel="schema.rdfs" type="text/html"
       href="http://www.w3.org/2000/01/rdf-schema#"/>
```

Figure 6 Use of Schema Handles

The following paragraphs would illustrate how these handles are used. We also describe how the mapping relations are encoded in HTML.

3.2 Encoding of the Mapping Relations In the Body Tag

The alignment of concepts technique that we use requires six mapping relations used in three design patterns. As stated earlier the mapping relations are: *owl::equivalentClass*, *owl::sameAs*, *rdfs::subclassOf*, *hasMatch*, *hasContext*, *hasRelation*. The e*quivalentClass* and *sameAs* relations are defined by the OWL specification. The *subclassOf* relation is defined by the RDFS specification. The remaining relations were introduced in the ontology alignment technique that we have implemented in our previous work. In this section we define the encoding of the mapping patterns in HTML. For all of the relations above, we used the *class* attribute of *DL* tag in combination with the *anchor*, *DT* and the *DD* tags. The interpretation of the encoding is also addressed in the next section.

The encoding of the *equivalentClass* relations between two entities is shown in Figure 7. Note the use of the OWL prefix in the class attribute of the *DL* tag. This is the same prefix that was declared in the *rel* attribute of the *link* tag of Figure 6:

```
<dl class="owl-equivalentClass">
   <dt><a href="http://mitre.org/owl/1.1/AM#CallSign">AM#CallSign</a></dt>
   <dd><a href="http://mitre.org/owl/1.1/AO#CallSignName">AO#CallSignName</a></dd>
</dl>
```

Figure 7 Encoding of the owl::equivalentClass

The encoding of the *owl:sameAs* relation between two entities is similar to that of the *owl:equivalentClass*, and is shown in Figure 8.

```
<dl class="owl-sameAs">
    <dt><a href="http://mitre.org/owl/1.1/AM#A10A">AM#A10A</a></dt>
    <dd><a href="http://mitre.org/owl/1.1/AO#A010A">AO#A010A</a></dd>
 </dl>
```

Figure 8 Encoding of the owl::sameAs

Next, we demonstrate the encoding of the *hasMatch* and *hasContext* relations. The first example shown in Figure 9 specifies that the triple AM AircraftType *hasMatch* the AO AircraftType. The second example

shown in Figure 9 specifies the triples AOCoord *hasContext* Coord-GEODETIC-WGE, and AMCoord *has-Context* Coord-UTM-WGE.

```
<dl class="mappings-hasMatch">
    <dt><a href="http://mitre.org/owl/1.1/AM#AircraftType">AM#AircraftType</a></dt>
    <dd><a href="http://mitre.org/owl/1.1/AO#AircraftType">AO#AircraftType</a></dd>
</dl>
<dl class="mappings-hasContext">
    <dt><a href="http://mitre.org/owl/1.1/AO#AOCoord">AO#AOCoord</a></dt>
    <dd><a href="http://mitre.org/owl/1.1/position#Coord-GEODETIC-WGE">
              position#Coord-GEODETIC-WGE</a></dd>
    <dt><a href="http://mitre.org/owl/1.1/AM#AMCoord">AM#AMCoord</a></dt>
    <dd><a href="http://mitre.org/owl/1.1/position#Coord-UTM-WGE">
              position#Coord-UTM-WGE</a></dd>
</dl>
```

Figure 9 Encoding of the hasMatch, and hasContext

Next, we present the encoding of the subclass of relation to specify that the aircraft A10A is a subclass of aircraft type. This is shown in Figure 10.

```
<dl class="rdfs-subclassOf">
    <dt><a href="http://mitre.org/owl/1.1/AM#A10A"></a></dt>
    <dd><a href="http://mitre.org/owl/1.1/AM#AircraftType"></a></dd>
</dl>
```

Figure 10 Encoding of the subclassOf

Finally, the *hasRelation* mapping relation, shown in Figure 11, is encoded in HTML to specify that that the entity Coord-UTM-WGE has two generic relations with UTM, and WGE. A generic relation is a genetic property where the name is not significant.

```
<dl class="mappings-hasRelation">
    <dt><a href="http://mitre.org/owl/1.1/position#Coord-UTM-WGE"></a></dt>
    <dd><a href="http://mitre.org/owl/1.1/position#UTM"></a></dd>
    <dt><a href="http://mitre.org/owl/1.1/position#Coord-UTM-WGE"></a></dt>
    <dd><a href="http://mitre.org/owl/1.1/position#WGE"></a></dd>
</dl>
```

Figure 11 Encoding of the hasRelation

4.0 Parsing of the WMSL and the Automatic Generation of the Aligned Ontologies

In our previous work, we have demonstrated that given the aligned ontologies of Figure 2, we can automatically translate an instance of the air mobility to an instance of air operations. Hence, what we'd like to show in this paper is that the aligned ontologies of Figure 2 can be generated from the WSDL files and the WMSL-Profile. Since, the WSDL files may not contain all the entities necessary to enable the information exchange; we use the mappings in the WMSL to specify, or create, the missing semantics. The end result is that the parsing of the WMSL yields the aligned ontologies with each of the ontologies corresponding to a WSDL file. (For now, we will ignore the case where the schema declared in the WMSL may themselves be ontologies.) A WMSL user agent accomplishes the generation of the ontologies as follows:

1. For each of the WSDL file declared in the WMSL-Profile, create its ontology.
2. For each mapping relation in the WMSL document, create the same relation between ontologies. If an entity is declared in the WMSL document, but is not present in the ontology, it is created before its relationship is asserted.

4.1 Generating Ontologies from WSDL Files

We start building the ontology by leveraging the semantics that already exist in the WSDL file. To do that, we create matching patterns between XML schema primitives and the OWL/RDF vocabulary; we derive class membership from XML schema sequence (or *xs:sequence*), and restrictions on properties from the minOccurs/maxOccurs attributes of the *xs:sequence* tag. Figure 12 shows a snippet of xml schema and Figure 13 shows the corresponding ontology of it. Since XML schema does not contain property names, we use a generic relationship in the RDF triple. This is completely justified since our previous work demonstrated that the property name of the triple within an ontology does not play a role in the reasoning necessary to reconcile syntactic, structural, and representational mismatches between data models.

```
<xs:complexType name="AircraftConfigType">
        <xs:sequence>
                <xs:element name="AircraftType" type="xs:string"
                                                minOccurs="0" maxOccurs="1"/>
                <xs:element name="CallSignName" type="xs:string"
                                                minOccurs="0" maxOccurs="1"/>
        </xs:sequence>
```

Figure 12 Snippet of XML Schema

```
<owl:Class rdf:ID="AIRCRAFTCONFIG">
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#HAS-AIRCRAFTTYPE"/>
      <owl:minCardinality rdf:datatype="&xsd;string">0</owl:minCardinality>
      <owl:maxCardinality rdf:datatype="&xsd;string">1</owl:maxCardinality>
      <owl:allValuesFrom rdf:resource="#AIRCRAFTTYPE"/>
    </owl:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <owl:Restriction>
      <owl:onProperty rdf:resource="#HAS-CALLSIGNNAME"/>
      <owl:minCardinality rdf:datatype="&xsd;string">0</owl:minCardinality>
      <owl:maxCardinality rdf:datatype="&xsd;string">1</owl:maxCardinality>
      <owl:allValuesFrom rdf:resource="#CALLSIGNNAME"/>
    </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

Figure 13 OWL Statements Corresponding to the XML Schemas of Figure 12

After the ontologies have been generated from the WSDL files, we proceed to augment these ontologies with semantics from the mapping relations in the WMSL- Profile.

4.2 Augmenting the Ontologies with Semantics from the WMSL Microformat

To illustrate this step, we proceed by generating the semantics from the mappings that were presented in the previous sections. The *equivalentClass* relation that was presented in Figure 7 yields the following OWL (Figure 14):

```
<owl:Class rdf:ID="CALLSIGNNAME">
        <owl:equivalentClass rdf:resource="&AM;CALLSIGN"/>
</owl:Class >
```

Figure 14 OWL Statements Corresponding to the WMSL of Figure 7

The *sameAs* mapping relation and the *subclassOf* called mapping relation that were presented in Figure 8 and Figure 10 respectively, yield the following OWL in the AM ontology and the AO ontology (Figure 15):

```
<owl:Class rdf:ID="A10A">
        <rdfs:subClassOf>
                <owl:Class rdf:about="&AM;AIRCRAFTTYPE"/>
        </rdfs:subClassOf>
        <owl:sameAs rdf:resource="&AO;A010A" />
</owl:Class>

<owl:Class rdf:ID="A010A"/>
```

Figure 15 OWL Statements Corresponding to the WMSL of Figure 8

The *hasRelation* mapping relation shown in Figure 11, yield the following OWL (Figure 16):

```
<owl:Class rdf:ID="Coord-UTM-WGE">
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#HAS_RELATION"/>
                        <owl:someValuesFrom rdf:resource="&position;UTM"/>
                </owl:Restriction>
        </rdfs:subClassOf>
        <rdfs:subClassOf>
                <owl:Restriction>
                        <owl:onProperty rdf:resource="#HAS_RELATION"/>
                        <owl:someValuesFrom rdf:resource="&position;WGE"/>
                </owl:Restriction>
        </rdfs:subClassOf>
</owl:Class>
```

Figure 16 OWL Statements Corresponding to the WMSL of Figure 10

Note that the OWL above is inserted in a position ontology that was declared by the WMSL-Profile. In similar fashion, the remaining mapping relations yield OWL definitions. When the user agent finishes the parsing of the WMSL-Profile, the aligned ontologies would have been created as shown in the Figure 2 above.

5.0 Relation to the Literature and Next Steps

These ideas draw on the proliferation of semantic matching techniques found in Semantic Web Services. In this paper, we abstract one such technique and present it in HTML. In the process, we demonstrated how WMSL is used to leverage existing schemas to produce ontologies. This allows us to think of WMSL as the glue between schemas and ontologies. WMSL can potentially enable matching between schemas irrespective of their formalisms. Today, techniques to embed semantics in HTML are emerging, but with a different purpose than WMSL. For example, the hcard Microformat is used to embed contact information in HTML pages. RDFa [15] serves to embed metadata such as those defined by the Dublin Core, in HTML. In contrast to RDFa, WMSL embed mapping relations in HTML. Another key distinction between the approach presented here and the Microformats is that WMSL builds on schemas, and not text pages. Moreover, the embedding of the mapping relations in HTML, serves to promote crosswalks for the purpose of building ontologies. This is a key difference from the tagging phenomenon that is so relevant in Folksonomies, or the annotation technique enabled by SAWSDL. That is, crosswalks may prove as significant to the structured data sources, as tags are to resources. Furthermore, since anyone can publish WMSL for existing WSDLs, we conclude that WMSL enables an open source model for building ontologies.

In conclusion, this paper describes how metadata in the form of mapping relations are embedded in HTML. We also described the parsing convention of WMSL by a user agent. Our next steps are to demonstrate how the mapping relations abstract workflow composition, and to make available libraries enabling the execution of WMSL webpages.

References

[1] Sabbouh, M., Higginson, J., Semy, S., Gagne, D. Web Mashup Scripting Language. Available at : http://semanticweb.mitre.org/wmsl/wmsl.pdf

[2] Webservice Description Language (WSDL) 1.1, http://www.w3.org/TR/2001/NOTE-wsdl-20010315, June 2006

[3] Gagne D., Sabbouh M., Powers S., Bennett S. Using Data Semantics to Enable Automatic Composition of Web Services. IEEE International Conference on Services Computing (SCC 06), Chicago USA. (Please see the extended version at:http://tinyurl.com/28svgr)

[4] Sabbouh M. et al. Using Semantic Web Technologies to Enable Interoperability of Disparate Information Systems, MTR: http://www.mitre.org/work/tech_papers/tech_papers_05/05_1025/

[5] Schroeder, B., & Sabbouh, M. (2005). Geotrans WSDL, http://www.openchannelfoundation.org/orders/index.php?group_id=348, The Open Channel Foundation

[6] Web Ontology Language (OWL), World Wide Web Consortium, http://www.w3.org/2004/OWL

[7] Resource Description Framework (RDF), World Wide Web Consortium, http://www.w3.org/rdf/

[8] More information on Microformats available at: http://microformats.org/

[9] More information on Asynchronous Javascript and XML (AJAX) available at: http://www.ajaxmatters.com/

[10] Really Simple Syndication (RSS) 2.0 specification, available at: http://www.rssboard.org/rss-specification

[11] Fielding, R.T., Architectural Styles and the Design of Network-based Software Architectures, PhD Dissertation in Information and Computer Science, 2000, available at: http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[12] McIlraith S., &Son T.,Zeng H. (2001). Semantic Web services. In IEEE Intelligent Systems (Special Issue on the Semantic Web)

[13] OWL-S: Semantic Markup for Web Services, http://www.w3.org/Submission/OWL-S/, November, 2004

[14] Web Service Modeling Ontology (WSMO), http://www.w3.org/Submission/WSMO/, June 2005

[15] RDFa Primer 1.0, available at: http://www.w3.org/TR/xhtml-rdfa-primer/