The Web Mashup Scripting Language Profile

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

The MITRE Corporation
202 Burlington Rd.
Bedford, Massachusetts 01730
ms@mitre.org

Abstract. This paper provides an overview of the Web Mashup Scripting Language (WMSL) and discusses the WMSL-Profile. It specifies the HTML encoding that is used to import Web Service Description Language (WSDL) files and metadata, in the form of mapping relations, into a WMSL web page. Furthermore, the WMSL-Profile describes the conventions used to parse the WMSL pages. It is envisioned that these WMSL pages scripted out by end-users using an easy-to-use editor will allow mashups to be created quickly to integrate Web services. The processing of these WMSL pages will be accomplished automatically and transparently to generate aligned ontologies sufficient for interoperability.

Key Words: Semantics, Scripting, Ontologies, Mapping Relations, Web Services, Mashup

1 Introduction

The Web Mashup Scripting Language (WMSL) [1] enables an end-user ("you") working with a browser—not even needing any other infrastructure, to quickly write mashups that integrate any 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 other WMSL web pages, but also affect a new programming paradigm that abstracts many programming complexities from the script writer. Furthermore, the WMSL web pages written by disparate end-users ("you") can be harvested by crawlers to automatically generate the concepts needed to build aligned ontologies sufficient for interoperability [4].

Despite many advances in Semantic Web technologies, such as OWL [6] and RDF [7], we have not observed widespread adoption of these technologies that was once anticipated. In comparison, a number of lightweight technologies such as Microformats [8], Ajax [9], RSS [10], and REST [11] enjoy substantial momentum and support from Web communities. The apparent reason for the success of these technologies is that they are effective in addressing needs and fairly simple to use.

We believe the adoption of Semantic Web technologies has been slow largely because they involve heavyweight infrastructure and substantial complexities. Adding to these issues are the multiple competing standards in Semantic Web Services [12],

e.g. OWL-S [13] and WMSO [14], and how they can be harmonized with existing W3C standards. Therefore, one key issue to address is this: Can we adopt Semantic Web technologies in a manner sufficient to our needs that is lightweight and much less complex for Web communities to use? Our previous research has clearly indicated that light semantics are sufficient in many applications, and can be used as a transitional step to rich semantics. For example, we concluded in [3] 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.

Another challenge to widespread adoption of rich semantics is the lack of social processes for the design of ontologies as is in the case for Folksonomies or in the social tagging case. Despite these difficulties, 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. How can we promote Web user participation in using semantics without requiring deep understanding of ontology?

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, and can be run from a browser. Also, our solution enables a light SOA approach where anyone can write a WMSL script to implement a mashup in support of information sharing requirements. Finally, WMSL can automatically generate the semantics needed to index and search structured data as is done with free text today.

In the next section, we provide an overview of WMSL. In section 3, we use an example to describe the encoding conventions of WMSL followed by its parsing conventions in section 4. In section 5, we relate this approach to the literature, discuss its implications, and point out the next steps to conclude the paper.

2 WMSL Overview

The WMSL script is divided into four blocks to contain different types of statements:

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
4. Mediation statements

Each of these blocks can be encoded either in HTML or a script. For the purpose of this paper, we discuss the WMSL-Profile: the encoding of the import and alignment blocks in the 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 pages are created by end-users with the help of an easy-to-use editor, and the parsing of the WMSL pages, which yields the aligned ontologies, is accomplished automatically and transparently.

Figure 1 shows a WMSL page for a use case 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>
```

Fig. 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. To accomplish this integration a WMSL page will be created. As stated earlier we will be focusing only on the import and alignment blocks. 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. These imports yield the aligned ontologies necessary to reconcile syntactic, structural, and representational mismatches between the AM and the AO schemas as was demonstrated in [3] and [4]. Moreover, these imports also yield the ontological description of web services necessary for their

automatic invocation and for handling their response—that aspect will not be addressed here but in a future paper.

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]. Our previous work [3] and [4] provides us a basis and insight to identify a minimal set of mapping relations for reconciling mismatches between data models in most cases if not all. The mapping patterns which are discussed in greater detail in our previous work are shown in Figure 2. The minimal set includes only these mapping relations:

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

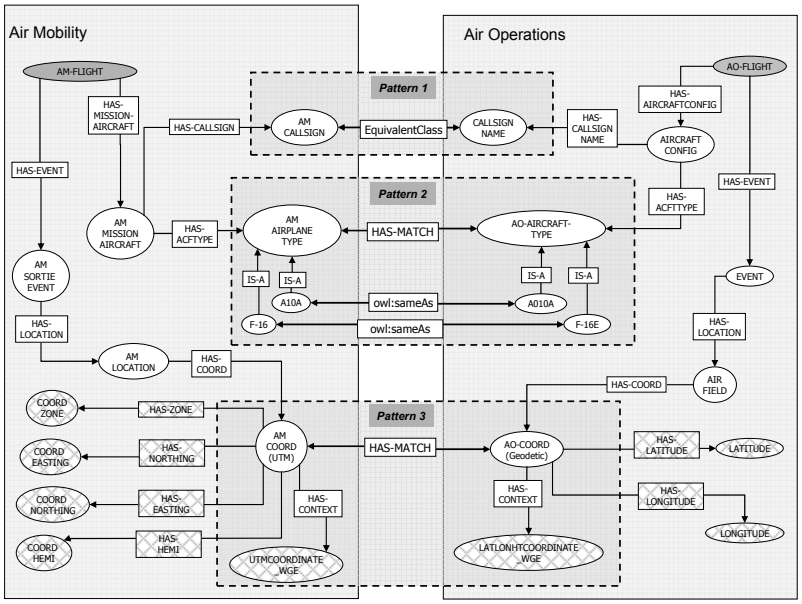

Fig. 2. Aligned Ontologies for the AM-AO Use Case

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. To conclude this section we highlight the fact that the imports of the WSDL files and the existence of these mapping relations in the WMSL enable an open-source/collaborative model of building aligned ontologies sufficient for interoperability.

3 WMSL-Profile Specifications

3.1 Encoding of the Schema Declarations Using the Header Block

We now specify the conventions used in encoding the WMSL-Profile. Other encodings are certainly possible, and we would welcome help for defining better encoding scheme. We have chosen HTML in which to define WMSL because of its already widespread acceptance and familiarity to Web communities. And there is no need to introduce new syntax and tags when the familiar standard HTML tags would suffice for WMSL.

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>
```

Fig. 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#"/>
```

Fig. 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. Notice that we adopt the type value “application/rdf+xml” to indicate an ontology file.

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

Fig. 5. Use of the Type Attribute

Next, we specify the handles for using relations from the RDFS and OWL specifications. In the next section we will show how these handles are used.

```
<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#"/>
```

Fig. 6. Use of Schema Handles

3.2 Encoding of the Mapping Relations in the Body Tag

The technique used for alignment 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*. In this section we define the encoding of the mapping patterns in HTML. For these relations the *class* attribute of *DL* tag is used 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 *owl::equivalentClass* relation 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>
```

Fig. 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>
```

Fig. 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 *hasContext* 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>
```

Fig. 9. Encoding of the hasMatch and hasContext

The encoding of the *rdfs:subclassOf* relation to specify that the aircraft A10A is a subclass of AircraftType 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>
```

Fig. 10. Encoding of the rdfs: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>
```

Fig. 11. Encoding of the hasRelation

4 Automatic Generation of the Aligned Ontologies When Parsing the WMSL-Profile

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 (AM) to an instance of the Air Operations (AO). How can we obtain the aligned ontologies of Figure 2? The WSDL files specified in the import block of the WMSL-Profile can be converted into ontologies. However, the WSDL files may not contain all the entities necessary to enable the information exchange; hence, we use the mappings in the WMSL-Profile to specify, or create, the missing semantics. The end result is that the parsing of the WMSL-Profile yields the aligned ontologies sufficient for integration. (For now, we will ignore the case where the schema declared in the WMSL may themselves be ontologies.) More details on generating aligned ontologies are described in the next sections.

4.1 Generating Ontologies from WSDL Files

We start building the ontology by leveraging the semantics already existing in the WSDL file. To do that, we create mapping patterns between XML schema primitives and the OWL/RDF vocabulary; the complete set of patterns will be discussed in a future paper. For example, class membership is derived from the XML schema sequence (*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. From our previous work we found 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>
```

Fig. 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>
```

Fig. 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 Derived from Mapping Relations

This step is illustrated by generating the semantics from the mappings presented in the previous sections. The *owl: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 >
```

Fig. 14. OWL Statements Corresponding to the WMSL of Figure 7

The *owl:sameAs* and *rdfs:subclassOf* mapping relations, 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"/>
```

Fig. 15. OWL Statements Corresponding to the WMSL of Figure 8

The *hasRelation* mapping relation shown in Figure 11, yields 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>
```

Fig. 16. OWL Statements Corresponding to the WMSL of Figure 10

Note that the OWL above is inserted into a position ontology that was included in the WMSL-Profile. In similar fashion, the remaining mapping relations yield OWL definitions. When we finish the parsing of the WMSL-Profile, the aligned ontologies are created as shown in the Figure 2 above.

5 Relation to the Literature and Future Work

The ideas presented in this paper draw on the proliferation of semantic matching techniques found in Semantic Web Services [13] literature. We abstract one such technique and formulate 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 from that of 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 is designed to 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 differentiator from the tagging phenomenon that is so relevant in Folksonomies or the annotation technique enabled by SAWSDL. That is, crosswalks may prove to be 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 conventions of WMSL by a user agent. Our next steps, which will be described in a separate paper, are to demonstrate how the mapping relations abstract workflow composition and to make available libraries for enabling the execution of WMSL web pages.

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/