Web Mashup Scripting Language

Marwan Sabbouh
The MITRE Corporation
202 Burlington Road, M315
Bedford, MA 01730-1420
001+781-271-2986
ms@mitre.org

Jeff Higginson
The MITRE Corporation
Hanscom AFB, Bldg 1607
Bedford, MA
011+781-377-3189
higginso@mitre.org

Salim Semy
The MITRE Corporation
202 Burlington Road, M325
Bedford, MA 01730-1420
001+781-271-8429
ssemy@mitre.org

Danny Gagne
The MITRE Corporation
202 Burlington Road, M315
Bedford, MA 01730-1420
001+781-271-5124
dgagne@mitre.org

ABSTRACT
The Web Mashup Scripting Language (WMSL) 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 that disparate end-users ("you") write, can be harvested by Crawlers to automatically generate the concepts needed to build lightweight ontologies containing local semantics of a web service and its data model, to extend context ontologies or middle ontologies, and to develop links, or mappings, between these ontologies. This enables an open-source model of building ontologies based on the WMSL Web page or scripts that end users ("you") write.

Categories and Subject Descriptors
D.2.12 Interoperability- Data mapping, Distributed objects

General Terms: Standardization

Keywords: Semantics, Ontologies, Scripts, Web Services, HTML

1. INTRODUCTION
The Web is undergoing another fundamental shift, often referred to as Web 2.0. There are several factors underlying the Web 2.0 revolution. On the back end, data management systems are able to support instantaneous changes to the database model, and the automated migration of instances of a data model to its next version. On the front end, scripting languages are enjoying a renaissance on the Web with Ajax. From a software engineering perspective, among the benefits of the Web 2.0, is the end of the software life cycle as we know it, the use of the Web as a platform, highly interactive content, and rich user interfaces. From a content creation perspective, open source creation of content is a key driver of successful companies, and is resulting in new winners and losers in the marketplace.

WWW 2007, May 8–12, 2007, Banff, Alberta, Canada.
ACM 978-1-59593-654-7/07/0005.

Meanwhile, on the web, we are faced with increased challenges in adopting semantics. Technologies such as OWL and RDF have not enjoyed the wide adoption that was once anticipated. Service Oriented Architecture (SOA) approaches are losing to 'light" approaches based on Microformats, Ajax, and REST. We believe that our solution enables a light SOA approach where anyone can write a WMSL web page to implement a mashup in support of information sharing requirements, and to automatically generate the semantics needed enabling indexing and searching capabilities for structured data, just as they work for free text.

2. APPROACH
In our previous work [1,3], to enable the information flow between legacy systems or web services in the enterprise, we demonstrated how to create a third web service that implements this integration. The created web service implemented a common data model comprised of attributes that are shared between the legacy schemas or legacy services, and attributes that need to be migrated from the legacy schemas into the common data model. In this work, we demonstrate that the common data model and the integration web service can be automatically generated from the pair-wise mappings of legacy web services' data models, and from the mappings of these data models to their appropriate context [2]. In this case, the use of context signifies a web service that can bridge the difference between entities or attributes belonging to the legacy data models. For example, a time web service that translates between the various time zones is able to bridge time representational mismatches between time entities in the legacy data models. Furthermore, we plan to demonstrate that the encoding of the mapping relations in an environment that combines HTML and scripting, such as a browser environment, results in a novel programming paradigm that offers the following benefits:

- Simplifies the writing of mashups by defining an object type that:
  - Abstracts the orchestration of workflows
  - Abstracts the reconciliation of syntactic, structural, and representational mismatches between data models
  - Abstracts argument passing between methods
  - Abstracts member variable and method signatures
- Enables and automates an open source model for semantics generation and ontologies creation using six simple mapping patterns to align concepts between data models: owl::equivalentClass, owl::sameAs, rdfs::subclassOf, hasMatch, hasContext, hasRelation
- Supports indexing and searching of metadata, employing existing web standards

We call this programming paradigm the Web Mashup Scripting Language (WMSL). In the next section, we present a simple use case and its corresponding solution in WMSL.

3. SAMPLE WMSL SCRIPT

We presuppose the existence of the web service that consumes the title of an event and returns the geodetic coordinates of the event's location. In order to display this event location on MapQuest the geodetic location needs to be translated into address information. This simple use case is depicted in Figure 1 below. Figure 2 shows a typical WMSL script that implements this solution.

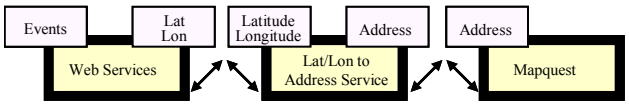

Figure 1: Sample Use Case

```
<html>
 <head>
    <title>WMSL Demo</title>
    <script type="text/javascript src="semantic.js"></script>
    <script language="JavaScript">
        <!-align entities→
        equivalentClass("lat", "Latitude");
        equivalentClass("lon", "Longitude");
        hasMatch ("GeodeticPosition", ""Address" );
        hasContext("GeodeticCoordinate","GeodeticPosition")
        hasRelation("GeodeticCoordinate", "WGE")
        sameAs("bos", "Boston")
     </script>
 </head>
 <body>
    <script language="JavaScript">
        <!—Import-->
        use ("http://semanticweb.mitre.org/contextontologies/position-ontology.owl");
        load ("http://www.mitre.org/events.wsdl");
        load ("http://www.mitre.org/ServiceTranslator.wsdl");
        load ("http://www.mitre.org/Mapquest.wsdl");

        <!--orchestrate workflow: invoke ReST style services in sequence, then mediate→
        step ("events-service", "GeodeticPosition", "EventsTitle");
        step ("Translator-service", "GeodeticPosition", "Address");
        step ("Mapquest", "Address");
    </script>
```

Figure 2: Sample WMSL Implementing the Use Case

In general the WMSL script contains four blocks: imports of Web Service Description Language (WSDL) [4] files, schemas, ontologies, and other WMSL scripts; alignments of entities and concepts; workflow statements; and mediation statements that can possibly be followed by other workflow statements. In figure 2, we demonstrate the use of the mapping relations. In total, we define six mapping relations that are used to align entities between schemas and to specify context. Not coincidentally, these are the same mapping relations that we have used in our previous work [1,3], 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 [3]. The mapping relations are: owl::equivalentClass, owl::sameAs, rdfs::subclassOf, hasMatch, hasContext, hasRelation. 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.

4. Related Technologies

We are seeing efforts on the Web to embed RDF in HTML [5]. However, it should be noted that our approach is different in the sense that we only need a small portion of the RDF vocabulary to enable this programming paradigm. Furthermore, to the best of our knowledge, we seem to be the first to establish that the mapping relations ease the writing of script code. Furthermore, one should note that we came upon this solution after having devised a semantic Web service approach. In that solution, we established that integration code in the form of a web services is generated from mapped ontologies. From that effort, we learned that the mapping relations were mostly responsible for the generation of the integration code. That is, the property name of a triple did not enter into the reasoning except when we were crossing ontologies. Hence, we define here the hasRelation mapping. We further established that only simple inferences, such as subsumption and class membership, were needed to enable our solution, and that the benefits of using owl and RDF were in specifying class definitions. All of these lessons were adopted in devising this solution. We refer the reader to the extended version of [1], for a more detailed discussion of the conclusions above.

5. References

[1] D. Gagne, M. Sabbouh, S. Powers, S. Bennett. 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)

[2] J. McCarthy. GENERALITY IN ARTIFICIAL INTELLIGENCE, Communications of the ACM, 30(12):1030-1035

[3] M. Sabbouh, 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/

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

[5] W3C Working Draft 16 May 2006. RDFa Primer 1.0: Embedding RDF in XHTML. http://www.w3.org/TR/xhtml-rdfa-primer/