# Harmonizing Semantic Web Services and Mashups Using WMSL

Marwan Sabbouh, The MITRE Corporation. 202 Burlington Road, Bedford, MA 01730
781-271-2964 (fax) ms@mitre.org
Jeff Higginson, The MITRE Corporation, Salim Semy, The MITRE Corporation, Caleb Wan, The MITRE Corporation, Danny Gagne, The MITRE Corporation

**Abstract – This paper harmonizes semantic web services and mashups by leveraging semantics in Schemas, and by generating aligned ontologies from mappings between schemas. Hence, we propose to encode the import of schemas and their mappings in HTML, which when combined with JavaScript libraries enable a web user to write mashups while hiding coding complexities and promoting reuse of schemas. These Web pages yield a Web of structured data similar to the web of unstructured text of today.**
**Keywords: Semantic Web Services, Mashups, Ontologies, Mapping Relations**

## 1.0 Introduction

Despite many advances in Semantic Web technologies, such as the development of multiple standards (RDF [18] /OWL [27]) to define semantics, we have not observed the widespread adoption of these technologies that was once anticipated. We believe there are a number of reasons for the lack of adoption of Semantic Web technologies:

- There is no common position in the Semantic Web community on how to adopt semantics. While some argue for domain ontologies that use mid-level and upper ontologies [7], others insist one define local semantics mapped to context ontologies [20].
- The Semantic Web community believes XML schemas do not provide a means to specify semantics and XML schemas are not a suitable starting point to capturing semantics. Hence, semantic standards, such as RDF and OWL, do not reuse schema primitives. While there is recent work to annotate [1] XML schemas with semantics, the process still requires the existence of well-defined ontologies.
- While ontologies formalize semantics for automated reasoning, data integration, and composition of web services, it's at the cost of introducing heavyweight infrastructure and exposing complexities to the user. That is, these techniques require the users to manually define ontologies.
- There are multiple competing approaches for achieving Semantic Web Services [11], e.g. OWL-S [17], WebML [26], WSDL-S [1] and WSMO [29].

Hence, Semantic Web promises remain years away from the widespread adoption.

In comparison, Web 2.0 technologies enjoy substantial momentum and support from Web communities because they facilitate web user participation and employ existing common standards and technologies. In contrast to the Semantic Web, the infrastructure for Web 2.0 is lightweight as the technologies are formed from existing standards and technologies. For example, Ajax uses a combination of client-side scripting, asynchronous HTTPRequest, HTML, and CSS, to enable Mashups for data integration of web services though active web user participation. Nonetheless, Web 2.0 also suffers from various limitations. For instance:

- Ajax libraries are complex and they result in mashups that have complex code.
- Mashups, such as those created by Yahoo Pipes [31], do not expose well defined data models, which makes them difficult to integrate with web services

or mashups created by other service providers.

In this paper, we propose the Web Mashup Scripting Language (WMSL) as an effective means to enable the Mashup of web services while specifying data semantics. WMSL is simply a web page with embedded metadata in the form of mapping relations [4, 15], imports of schemas, and scripting to achieve the mashing up of services. The existence of the mapping relations permits the creation of JavaScript [8] libraries that abstract the invocation of web services, the handling of their returned documents, and mediation between web services. Thus, WMSL utilizes existing standards and technologies, emphasizes the role of mapping relations and scripting, and encourages user participation to capture formal semantics. With each Mashup having a corresponding WMSL, it also enables integration of multiple Mashups via imports of and associations across WMSL files. Finally, WMSL can be processed by a user agent to generate aligned ontologies [16].

The remainder of this paper proceeds as follows: in the next section, we present a simple model to highlight the steps involved in creating a typical Semantic Web Service solution. In section 3, we use the same model to capture the steps involved in creating a Mashup. In section 4 and 5, we summarize the Web Mashup Scripting Language and conclude the paper.

## 2.0 Semantic Web Services

The composition of web services [12,13] is achieved by creating a third web service (hereafter referred to as the integrating web service) and its Web Service Description Language (WSDL) [28] file. This web service invokes the source legacy web service to retrieve data, mediates to resolve any mismatches the data has with the destination web service, typically though invocation of context [9, 10] services, and then passes the converted data to this destination web service. This integration model is illustrated in Fig. 1.

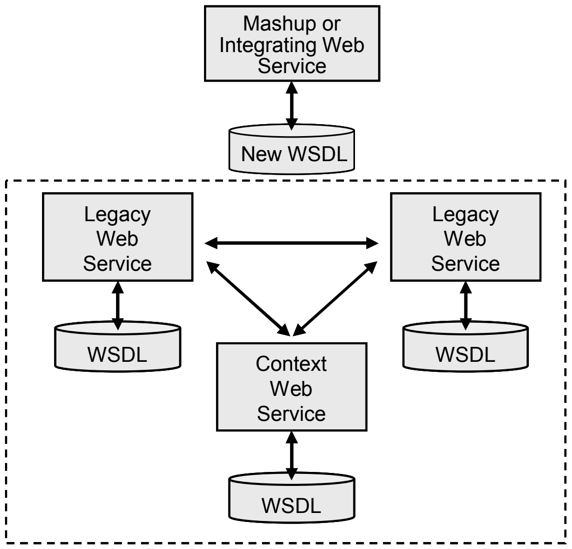


**Figure 1. Model for Integrating Web Services**

Fig. 1 shows the integration of two legacy web services through the use of a third integrating service. Typically, the integrating service's WSDL file is very similar to the destination service's WSDL with the addition of (few) entities from the source's and context's WSDL files.

The typical SWS approach [20, 5] presented here accomplishes the automatic composition of web services by representing the web services ontologically. Typically, a web service has the following ontological components:

1. Description of web services as an OWL-S upper ontology [17]
2. Representation of WSDL ontologically [20]
3. Shared ontologies describing the domain or the context of the data being exchanged
4. Mappings between WSDL ontologies and the shared ontologies to annotate WSDL files [1]
5. Ontology of mapping relations to map the WSDL ontologies to the shared ontologies and the shared ontologies to the OWL-S upper ontologies

These ontology components and the relationships among them are illustrated in Fig. 2.

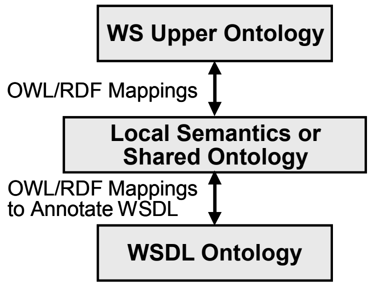

WS Upper Ontology
OWL/RDF Mappings
Local Semantics or Shared Ontology
OWL/RDF Mappings to Annotate WSDL
WSDL Ontology

**Figure 2. Ontological Representation of Web Services**

There are two types of mappings, as depicted in Fig. 2, needed to represent a Web service ontologically. The first is used to map a WSDL ontology [20] to the shared ontology, for example annotating a WSDL file with concepts from a shared ontology. The second set of mappings is needed to integrate a Web service into a shared ontology. These mappings are accomplished by creating links, or mapping primitives, between two ontologies.
Once ontologies are created describing each web service, one can achieve their integration by mapping concepts of each ontology to each other and to context ontologies [5]. Fig. 3 presents an ontological model corresponding to the model shown in Fig. 1.

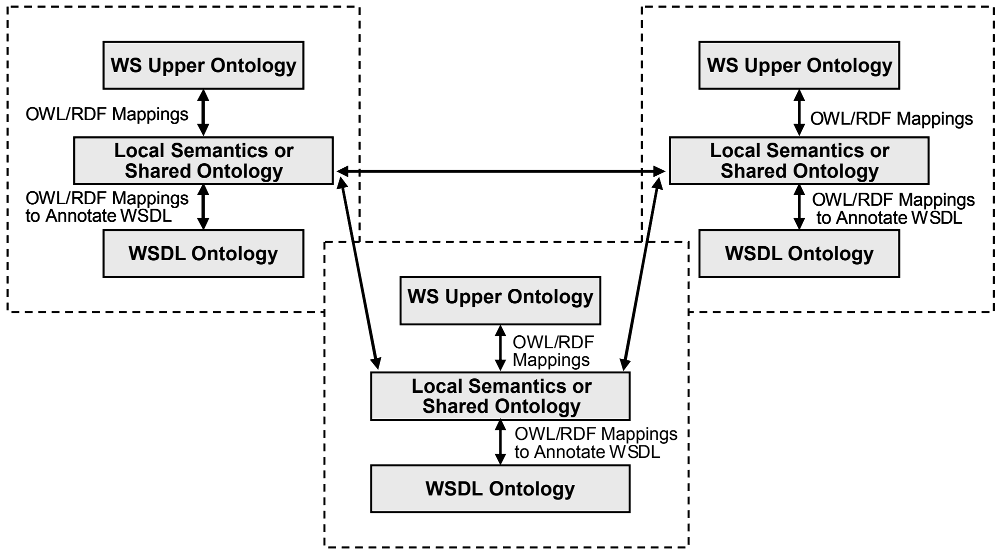

WS Upper Ontology
OWL/RDF Mappings
Local Semantics or Shared Ontology
OWL/RDF Mappings to Annotate WSDL
WSDL Ontology
WS Upper Ontology
OWL/RDF Mappings
Local Semantics or Shared Ontology
OWL/RDF Mappings to Annotate WSDL
WSDL Ontology
WS Upper Ontology
OWL/RDF Mappings
Local Semantics or Shared Ontology
OWL/RDF Mappings to Annotate WSDL
WSDL Ontology

**Figure 3. Ontological Mapping of Shared Ontologies**

The mappings illustrated in Figs. 2 and 3 are important in representing both web services ontologically and to reconcile syntactic, structural, and representation mismatches between the legacy web services. There are some interesting properties to highlight about the mappings. First, the mappings relate entities between ontologies. Second, the mapping of entities results in the duplication of entities from one ontology to the other. Third, entities with representational mismatches, e.g Inches to cm, are not only mapped pair-wise, but are also mapped to a Context [9, 10] that reconciles representational mismatches. The duplication of entities is illustrated in Fig. 4.

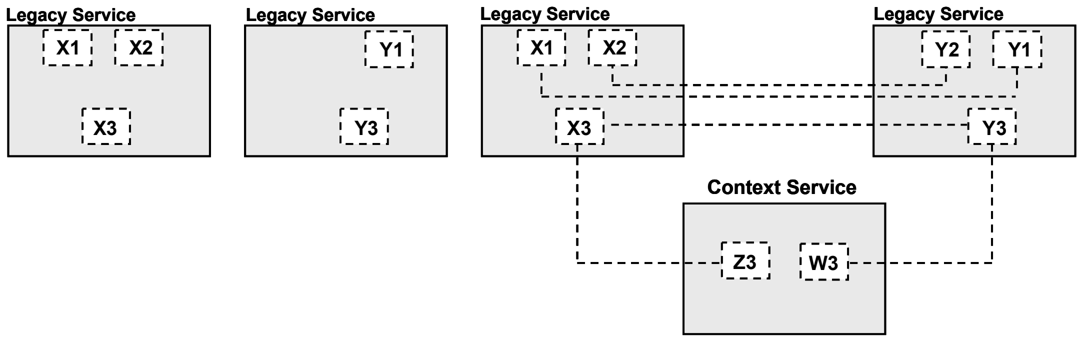


**Figure 4. Mapping Properties**

Fig. 4 shows the X2 attribute from the source service is duplicated as Y2 in the destination service. It also shows attributes X3, and Y3 being mapped to a context service.

Once the mappings are done, reasoning over the mapped ontologies results in code generation [5]. In addition, we postulate the WSDL file associated with the integrating service can also be generated from the mapped ontologies. Work is in progress to validate this hypothesis.

To invoke web services automatically, we have to reason over the mappings between the WSDL ontology and the shared ontologies. To handle the return documents of invoked services, we reason between the WSDL ontologies and one or more shared ontologies. To reconcile structural, syntactic, and representational differences, we reason over the mappings between the shared ontology of legacy services and context services.

It turns out the key to reasoning algorithms are graph traversal techniques which make use of simple inferences and rely on the following mapping primitives and relationships [20, 5]:

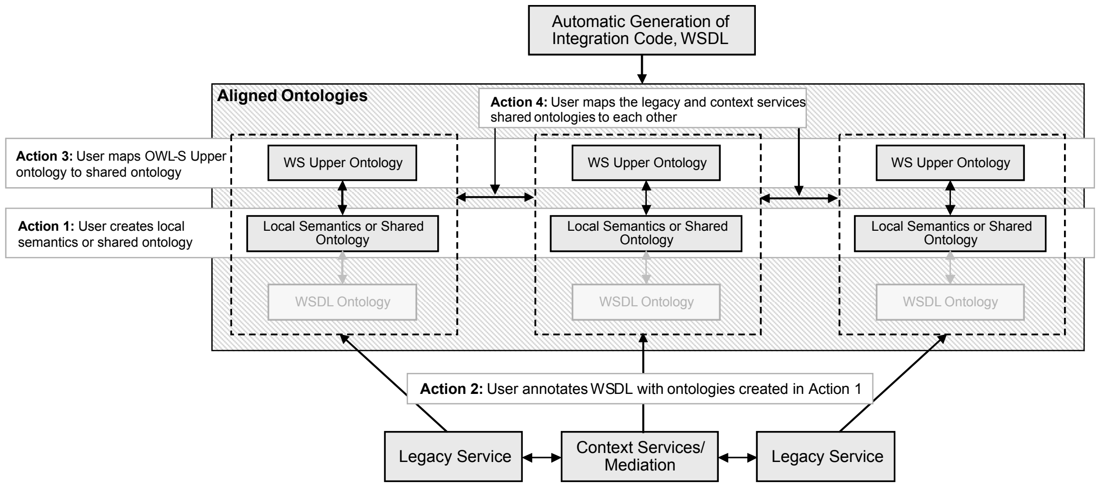
- A primitive to define generic relationships to implement the duplication of entities are required by the mapping technique
- Two specialized relationships, i.e. has-Match and has-Context, to implement the mapping of entities with representational mismatches
- The inferences include: subclassOf, memberOf, sameAs, equivilantClass, inverseOf, and transitivity relations

A key property of the graph traversal technique is the ability to exclude a relation's name from the graph traversal [20]. For example, one can say "navigate the graph from concept *this thing* to concept *another thing*, excluding relationship hasMatch".

Another point to note is that the name of the property between concepts in an ontology never enter into the reasoning process. In fact, the property name is only important when traversing between ontologies, or when the relationship is one of the mapping primitives, e.g. sameAs, equivalentClass

## 2.1 Discussion of Semantic Web Services Approach

It's clear that the SWS approach imposes new requirements and complexities on the development process to automate integration. These requirements are shown in Fig. 5.



Automatic Generation of Integration Code, WSDL
Aligned Ontologies
Action 4: User maps the legacy and context services shared ontologies to each other
Action 3: User maps OWL-S Upper ontology to shared ontology
Action 1: User creates local semantics or shared ontology
WS Upper Ontology
Local Semantics or Shared Ontology
WSDL Ontology
Action 2: User annotates WSDL with ontologies created in Action 1
Legacy Service
Context Services/ Mediation
Legacy Service

**Figure 5. Requirements on Web Users for Adopting a Typical SWS Solution**

At the center of this approach are ontological components. Many of the ontological components described above, illustrated in Fig. 5, require users to manually design ontologies, while others can be generated automatically. For instance, the WSDL ontology, (dimmed in Fig. 5), is generated automatically from WSDL files. The web service upper ontology, while designed manually for one web service, it may be reused for all other web services. In contrast, the shared ontology must be manually implemented as are the mappings between shared ontologies. These ontologies and mappings are dependent on particular web service domains.

In designing ontologies, there is lack of clarity, from a user perspective, on what defines a good ontology and the requirements placed on the ontology by the SWS approach. The typical SWS approach presented in this paper suggests that the design of the shared ontologies is not arbitrary. Rather, the mapping techniques used impose a methodology on the design of the shared ontologies. First and foremost, in order to annotate WSDL files with the shared ontology, all entities from the WSDL must be duplicated in the shared ontology. This is a direct consequence of the fact that the mapping technique results in duplication of entities. Therefore a subset of the shared ontology must correspond to the entities present in the WSDL file. Furthermore, the mappings between shared ontologies require data type information, such as instances or individuals of aircraft types, vehicle types, etc. These types must be explicitly specified in each of the ontologies. Finally, the mapping of ontologies is clearly facilitated by concepts that are ubiquitous in a large number of domains, such as geospatial information.

The formalism of choice to design the ontological components has been OWL and RDF. The use of OWL and RDF has primarily been based on the assumption that particular inferences are needed to accomplish the automated composition of web services and OWL and RDF provide these primitives and mappings to perform the inference.

However, as indicated above, the key to reasoning are graph traversal algorithms based on simple inferences.

## 3.0 Web 2.0

Web 2.0 technologies enjoy substantial momentum and support from Web communities because they facilitate user participation and employ existing common standards and technologies. Web 2.0 is all about empowering the user, turning the web into a programmable environment where application programmatic interfaces and a minimal learning curve allow users to contribute information as well as exploit existing information in previously unimagined ways.

Mashups are a great example of Web 2.0 technologies. Mashups support integration and derivation of hybrid application by third parties, enabling novel forms or reuse. While original Mashups were limited in their scope, i.e. pair wise combinations with output typically being another website and the need for programming experience, emerging capabilities such as Yahoo Pipes are breaking this barrier. Yahoo Pipes generalizes mashups by providing a drag and drop graphical user interface to allow users to connect heterogeneous data sources, process them, and redirect the output to one of multiple applications. Mashups are emerging as a new paradigm for lightweight data integration.

While the impact of Web 2.0 technologies such as Mashups is profound, current implementations suffer from various limitations. The openness of web data sources and Mashups is limited to providing application programming interfaces (API) for developers to program against, falling short of enabling integration and reuse of data. Mashups, such as those created by Yahoo pipes, do not inherently expose well defined data models. Instead, the data models are hidden within application code, making reusability and extensibility difficult. While it is possible to develop data models associated with Mashups, the process of doing so is independent of the process to develop the Mashup and thus is duplicative.

A consequence of no associated data models is that Mashups created within a particular service provider environment, such as Yahoo Pipes, do not support integration of the resulting Mashup with other web services. The lack of exposed data models limits reuse of the Mashup to the person who initially developed the Mashup and has access to the application code. A third party that wishes to either extend, reuse, or compose existing Mashups does not have sufficient information, i.e. data models of the Mashups, to do this. Furthermore, the lack of standards to expose Mashup metadata does not support interoperability across Mashup service providers.

## 4.0 Harmonizing Semantic Web Services with Mashups: Web Mashup Scripting Language

The Web Mashup Scripting Language (WMSL) aims to harmonize the Semantic Web Services and mashups, while addressing the issues identified in the discussion on each approach above. First, we derive conclusions based on the key findings from the Semantic Web Service discussion. Then, we propose WMSL that implements these conclusions. Finally, we show that WMSL can benefit both the Semantic Web Services and Mashup communities.

The key findings from the Semantic Web Services discussion can be summarized as:

1. The annotation of XML schemas with shared ontologies result in the duplication or migration of entities between XML schemas and the shared ontology
2. The reasoning over mapped ontologies to automatically compose web services

requires only graph traversal algorithms which make use of a defined set of mapping primitives and simple inferences
3. The relationship name within an ontology does not play a factor in the reasoning to automate composition of web services
4. Context services and specifying type information play a critical role in enabling mappings between schemas

The above findings furthermore lead us to the following conclusions:

1. The first finding above highlights a need to reuse existing entities found in XML schemas in the design of shared ontologies. Furthermore, one can also leverage the schema primitives, such cardinality constraints and class definitions in building shared ontologies. Hence, there is a need to map XML schema's primitives to OWL/RDF primitives [22].
2. The second finding above suggests that the sets of mapping relations are sufficient to achieve the automation described above.
3. The third finding suggests that the mapping relations can operate directly on XML schemas. This is significant since without a need for the shared ontology, the ontology of mapping relations of figure 2 are no longer needed. Hence, there is a need only for the set of mapping relations between legacy and context services. Furthermore, this implies that ontologies can be hidden from the user, as aligned ontologies can be automatically generated from the XML schemas and the mapping relations.
4. The fourth finding suggests that a well defined schema is useful irrespective of the formalism used.

The above conclusions lead to a new approach of composing web services. This approach is referred to as the Web Mashup Scripting Language.

The Web Mashup Scripting Language (WMSL) [21] enables a web-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. In general the WMSL script contains four types of blocks [22]:

1. Imports of Web Service Description Language (WSDL) files [28], schemas, ontologies, and other WMSL scripts
2. Alignments of entities and concepts
3. Workflow statements
4. Mediation statement

First, the WMSL imports the WSDL files or the schemas of legacy and context web services. Then, the WMSL uses six mapping relations to align entities between the schemas. Not coincidentally, these are the same mapping relations that are used in our previous work [5, 20], with the exception that they are defined outside the ontologies. 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. The scripting statements that a Web-user writes are high level workflow statements, in addition to any custom coding that may be required. Note that these relations are not only used to match entities existing in XML schemas, but also they can be used to assert new entities that are not found in legacy XML schemas. The paper titled, WMSL-Profile [22], specifies the HTML encoding that is used to import WSDL files and the mapping relations, into a WMSL web page. Furthermore, the WMSL-Profile describes the conventions used to parse the WMSL pages by a WMSL user-agent. Currently we are in the process of defining the JavaScript object types needed and the application programming interface (API) in support of the workflow and mediation statements.

## 4.1 Benefits of WMSL

Web users accomplish the composition of web services by writing WMSL web pages. User agents crawl these web pages to generate aligned ontologies. This process is depicted in Fig. 6.

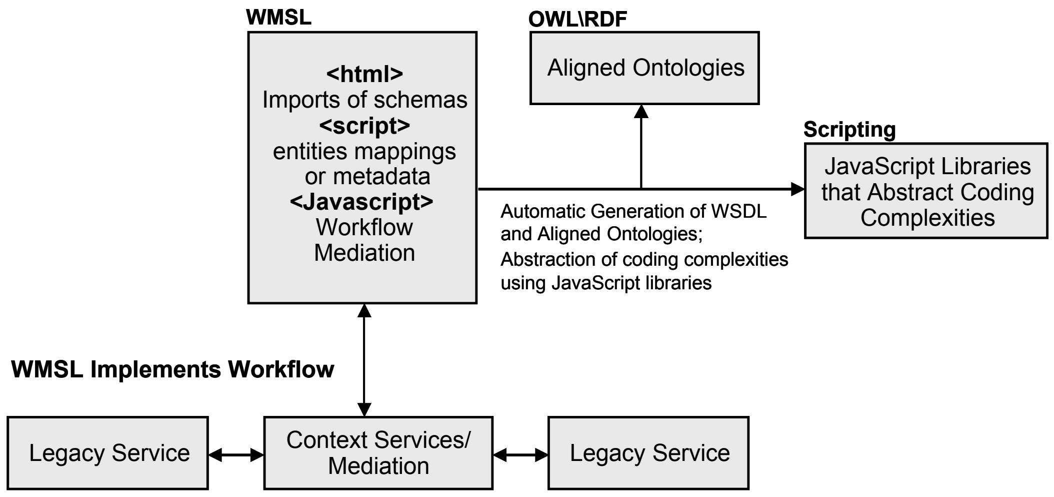

WMSL
<html>
Imports of schemas
<script>
entities mappings or metadata
<Javascript>
Workflow
Mediation
OWL\RDF
Aligned Ontologies
Scripting
JavaScript Libraries that Abstract Coding Complexities
Automatic Generation of WSDL and Aligned Ontologies; Abstraction of coding complexities using JavaScript libraries
WMSL Implements Workflow
Legacy Service
Context Services/ Mediation
Legacy Service

**Figure 6. WMSL Model of Adoption**

Fig. 6 demonstrates that the starting point for defining semantics is XML schemas, while ontologies result from the XML schemas and the WMSL. Therefore, the ontological complexities are hidden from Web users. Furthermore WMSL claims the following advantages:

- Since aligned ontologies are generated from WMSL web pages, it follows that WMSL enables an open-source/collaborative model of building aligned ontologies.
- Since we have shown that integration code can be automatically generated from aligned ontologies, we conclude that WMSL abstracts the mediation coding complexities from the scriptwriter. In addition to mediation, WMSL automates argument passing, web services invocation, and the handling of returned response of invoked services. We also suggest that WMSL can contribute to the state of the art in scripting by providing APIs for high-level workflows that introduce mediation and context object types to scripting paradigms.
- Since WMSL web pages can import other WMSL pages, we conclude that WMSL enables reuse of mashups created by different hosting providers.
- Since WMSL can import ontologies and XML schemas, this positions WMSL as the glue between different schema formalisms.

- Furthermore, assuming that WSDL files can be generated from aligned ontologies, we conclude that the WSDL of the mashups can be automatically generated from the WMSL web page.

## 5.0 Conclusion

WMSL has significant consequences on today's W3C standards.

- **WMSL uses XML schemas as a starting point for specifying semantics**

In particular, the relationship between the XML schema standard and the RDF/OWL standards needs to be revisited. XML schemas must be the starting point for specifying semantics which can be captured in ontologies by user agents. This can happen through a standardization that maps the XML schema primitives to that of the RDF/OWL primitives.

- **WMSL reassesses the role of SAWSDL**

Given that XML schema is the starting point for ontology development, then annotation between WSDL files and ontologies makes sense if ontologies are available to Web users. In the absence of such ontologies there is no need for annotation.

- **WMSL standardizes on a minimal set of mapping relations in HTML/XHTML for augmenting the semantics found in XML schemas**

We also call for the standardization of the minimal set of mapping relations that accomplish the composition of Web services. We favor a standard for encoding the mapping relations in HTML or XHTML, as HTML can be combined with scripting to run either on Web servers, or in browsers. Today, techniques to embed semantics in HTML are emerging, but with a different purpose than WMSL. For example, the hcard Microformat [14] is used to embed contact information in HTML pages. Another key distinction between the approach presented here and the Microformats is that WMSL builds on schemas, and not text pages. RDFa [19], eRDF serves to embed metadata such as those defined by the Dublin Core [30], in HTML. These approaches result in the generation of RDF, and not aligned ontologies. GRDDL [6] provides means to generate RDF from instance data which can then be queried using SPARQL [23]. However, GRDDL, as it is, does not define the mappings that we need between XML schema's primitives and those of RDF\OWL, and makes use of XSLT. Furthermore, note that the mapping relations in WMSL implement the mapping properties discussed above.

- **WMSL promotes crosswalks between schemas**

The embedding of the mapping relations in HTML, serves to promote crosswalks for the purpose of building aligned ontologies. This is a key differentiator from the tagging phenomenon that is so relevant in Folksonomies. That is, crosswalks may prove as significant to the structured data sources, as tags are to resources.

- **WMSL creates a web of mashups**

In contrast to the islands of mashups that are emerging today, the proposed approach gives birth to a web of structured data, or a web of mashups, effectively addressing the deep Web [3] problem. That is, the boundaries that exist today between mashups created by different hosting providers disappear with WMSL adoption.

- **WMSL is Scalable**

Finally, we postulate that this approach is scalable, since WMSL Web pages are

HTML and scripting. That is, from a global network topology prospective, we postulate that the emergent topology of the web of Mashup is similar to the topology of the current web [2]; a complex network [24] with small world properties [25]. Note that we aren't concerned that the scripting would affect scalability, since the import of WMSL pages can exclude scripting if needed.

## 6.0 Acknowledgments

This paper is the result of several years of work on the semantic web. We would like to thank Dave Lehman who, as CTO of MITRE, aggressively funded this area of research, and whose encouragements and support are invaluable to us in our daily engagements with our stakeholders. We are also grateful to Mr. Bert Hopkins for sponsoring our work throughout the years. Additionally we would like to thank those who supported, engaged, and collaborated with us the past few years, and whose contributions shaped this work.

## 7.0 References

[1] R. Akkiraju, J. Farell, J.A. Miller, M. Nagarajan, A. Sheth, and K. Verma, (2005) Web Service Semantics - WSDL-S, http://www.w3.org/2005/04/FSWS/Submissions/17/WSDL-S.htm

[2] A. Barabasi, Statistical Mechanics of Complex Networks. Rev Mod Phys 2002, 74, 47-97

[3] M. Bergman, The Deep Web: Surfacing Hidden Value, The Journal of Electronic Publishing, http://www.press.umich.edu/jep/07-01/bergman.html

[4] M. Crubezy, Z. Pincus, and M.A. Musen, (2003). "Mediating knowledge between application components", Semantic Integration Workshop of the Second International Semantic Web Conference (ISWC-03), Sanibel Island, Florida, CEUR, 82.

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

[6] GRDDL, Gleaning Resource Descriptions from Dialects of Languages, http://www.w3.org/TR/grddl/

[7] T.R. Gruber, (1993). "A translation approach to portable ontologies", J on Knowledge Acquisition, Vol 5(2), p199-220

[8] JavaScript (ECMA Script), http://www.ecma-international.org/publications/standards/Ecma-262.htm

[9] J. McCarthy, (1987),GENERALITY IN ARTIFICIAL INTELLIGENCE, Communications of the ACM, 30(12):1030-1035

[10] J. McCarthy, (1993), Notes on Formalizing Context., In Proceeding of The Thirteenth International Joint Conference on Artificial intelligence, Pages 555-560, Chambery

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

[12] B. Medjahed, and A. Bouguettaya, (2005) A Multilevel Composability Model for Semantic Web Services. IEEE Transactions on Knowledge and Data Engineering (TKDE)

[13] B. Medjahed, and A. Bouguettaya, (2005) A Dynamic Foundational Architecture for

Semantic Web Services. Distributed and Parallel Databases (DAPD), 17(2)

[14] Microformats, http://microformats.org/

[15] P. Mitra, G. Wiederhold, and M. Kersten, (2000). A Graph-Oriented Model for Articulation of Ontology Interdependencies"; Extending DataBase Technologies, EDBT 2000, Konstanz, Germany

[16] N. Noy, and A. Musen, (2000). PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In 17th National conference on Artificial intelligence. Austin, Texas

[17] OWL-S (2004). Semantic Markup for Web Services. http://www.w3.org/Submission/OWL-S

[18] Resource Description Framework (RDF). (2006). World Wide Web Consortium, http://www.w3.org/rdf/

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

[20] M. Sabbouh, J. Derosa, S. Powers, and S. Bennett, Using Semantic Web Technologies to Enable Interoperability of Disparate Information Systems, MTR: http://www.mitre.org/work/tech_papers/tech_papers_05/05_1025/

[21] M.Sabbouh, J. Higginson, S. Semy, D. Gagne, Web Mashup Scripting Language, Poster, WWW2007, Banff, Canada

[22] M. Sabbouh, J. Higginson, C. Wan, S. Semy, D. Gagne, The Web Mashup Scripting Language Profile, Proceedings of the European Semantic Web Conference, Workshop on Scripting the Semantic Web, Innsbruck 2007

[23] SPARQL, http://www.w3.org/TR/rdf-sparql-query/

[24] S. H. Strogatz. Exploring complex networks. Nature 410: 268-276 (2001)

[25] D. J. Watts, and S.H Strogatz, Collective Dynamics of Small World Networks, Nature 1998, 393, 440-442

[26] WEBML, Flexible Specification of Semantic Services using Web Engineering Methods and Tools, In Proceedings of the SWESE 2006, ISWC 2006, Athens, GA, USA, November 2006.

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

[28] Web Service Description Language (WSDL) 1.1. (2006). http://www.w3.org/TR/2001/NOTE-wsdl-20010315

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

[30] S. Weibel, T. Koch, The Dublin Core Metadata Initiative, D-Lib Magazine, 2000

[31] Yahoo Pipes, http://pipes.yahoo.com/pipes/