Ontology Driven Data Mediation in Web Services

"Extended and invited from ICWS 2006 with Id# 174"

Meenakshi Nagarajan, Kunal Verma, Amit P. Sheth, John A. Miller
LSDIS Lab, Department Of Computer Science, University of Georgia, Athens GA, USA
{bala,verma,sheth,jam}@cs.uga.edu

ABSTRACT:

With the rising popularity of Web services, both academia and industry have invested considerably in Web service description standards, discovery, and composition techniques. The standards based approach utilized by Web services has supported interoperability at the syntax level. However, issues of structural and semantic heterogeneity between messages exchanged by Web services are far more complex and crucial to interoperability. It is for these reasons that we recognize the value that schema/data mappings bring to Web service descriptions. In this paper, we examine challenges to interoperability; classify the types of heterogeneities that can occur between interacting services and present a possible solution for data interoperability using the mapping support provided by WSDL-S, a key driver behind SAWSDL. We present a data mediation architecture using the extensibility features of WSDL and the popular SOAP engine, Axis 2.

KEYWORDS:
Data Mediation, Ontology, Web Service interoperation, Message-level heterogeneities, Matching, Mapping, SAWSDL, METEOR-S, Axis 2.0

1. Introduction

The emergence of Web services and service oriented architectures is leading to new innovative enterprise solutions based on composition of Web services to realize business and scientific processes. So far, much of the research has focused on discovery (UDDI), composition (Sivashanmugam, 2004b), (Medjahed, 2003), (Zeng, 2003) and execution (BPEL) of Web services. One of the biggest stumbling blocks in the grand vision proposed by SOA is data heterogeneity between interoperating services. By data or message level heterogeneities, we refer to incompatible formats of messages exchanged by the services. This is not a new problem. Since the inception of federated databases (Sheth, 1990), interoperability among databases with heterogeneous schemas has been a well researched issue (Litwin, 1986) (Sheth, 1998). In this paper, we discuss message level heterogeneities in the Web services domain and present an approach for resolving these heterogeneities. This work was done as a part of the METEOR-S project, which defines four kinds semantics for Web services - data, functional, non-functional and execution semantics and utilizes these semantics in the complete lifecycle of semantic Web processes (publishing, discovery, composition, execution and monitoring).

Typically enterprise systems are developed over several periods of time, by diverse organizations and not necessarily with the same structures and vocabularies. This leads to substantial heterogeneity in syntax, structure and semantics when it comes to interoperation between these systems. For example, one system may encode performance as grades A-F, while another may use scores ranging from 1-100. A recent approach to interoperate between such systems exposed as Web services has been semantically representing the functional capabilities of the services and then using semantic discovery techniques to find and compose these services into a process. A common fallacy of such an approach is the assumption that a semantic match ensures interoperation.

To appreciate this, consider the case of a process that uses two Web services with heterogeneous message schemas (i.e., the input and output message schemas are incompatible) and the output of the first service is supplied as an input to the second service. The process of resolving these heterogeneities and transforming one message format to another is also referred to as data mediation. A simple solution to achieve data mediation between the services is to manually create a mapping from the first service's output to the second service's input (this is the proposed solution of most enterprise integration products in Web services). However, this mapping would have to be created every time services in the process are changed or upgraded, potentially making the number of generated mappings very large. An alternate solution to this problem (which is the approach we use) is mapping the inputs and outputs of the services to a conceptual model and using those mappings for interoperating between the services.

In this paper, we classify impediments to data interoperability among Web services by adapting previous work on semantic interoperability in databases (Kashyap, 1996). Our approach uses the support for data mapping provided by SAWSDL (SAWSDL), an upcoming candidate recommendation for semantic annotations for WSDL by a W3C chartered working group. Semantic Annotations for WSDL (SAWSDL) is an effort to define mechanisms by which semantic annotations can be added to WSDL components. Many of the concepts in SAWSDL are based on an earlier effort WSDL-S (WSDL-S), a W3C submission. This work was conducted as a part of WSDL-S before the formation of the SAWSDL group. In this paper, we will limit our discussions of SAWSDL as relevant to this work.

The aim of this work is to provide a solution to the problem of Web service interoperation by making incremental changes to Web services tools. Since SAWSDL builds upon existing Web services standards (WSDL), it also allows us to use the extensibility support provided by Axis 2 to implement the data mediation. This paper has the following contributions:

- We present a comprehensive, practical approach for resolving data heterogeneities between Web services using semantic domain models, i.e., Ontologies.
- We adapt previous work on schema and database integration to compile different kinds of heterogeneities one might encounter during the interoperation of Web services.
- We present a data mediation architecture that is built using the extensible elements of existing Web service standards (WSDL) and tools (Axis 2).

The rest of this paper is organized as follows. We motivate the need for resolving data mediation in Web services with an example in Section 2; classify the possible data or message-level heterogeneities that can occur in real world interoperable services in Section 3; briefly describe the data mediation support provided by SAWSDL with examples in Section 4; present our approach, system architecture and implementation details in Section 5 and 6; evaluate the need for data mediation in Section 7; illustrate the use of our comprehensive approach in the runtime plug-n-play of services in a Web process in Section 8; compare our work with other related efforts in Section 9 and conclude with discussions on the challenges we faced and our experiences in this space in Section 10.

2. Motivating Scenario

To elucidate the need for data interoperability, we present a simple use case using two real-world Web services. Consider the process of an auto company that sends customers special offers and coupons by mail using the phone numbers that customers provide at the time of purchase. The process consists of making calls to two Web services, each from different providers, to get the information that its marketing analyst needs. The first Web service is a directory listing Address Lookup service available at (PhoneLookupService) that returns an address for a listed telephone number. The second Web service available at (GeocodeService) is a Geocode Enhancer service that uses an address to provide demographic and logistical information. The collective data from the services is used by the client to make strategic marketing decisions. The only problem is that the output of the Address Lookup service is not compatible with the input required by the Geocode Enhancer service. Figure 1 shows the process composed using the two Web services and the message elements exchanged.

There are two conceivable approaches to solving this problem. The first involves using custom rules or mappings to transform the output of the first service to the input of the second service. However, in the event that the auto company decides to change any of the services, it would have to construct these manual mappings again. The second approach involves providing mappings to a generic domain model and utilizing it to do the conversion of messages. This gives the ability to plug 'n' play services from different providers as long as they also provide mappings to the same domain model. If two services provide mappings to different domain models, mappings between the domain models can be used to facilitate interoperation between services. The rest of this paper discusses a possible solution using the latter approach.

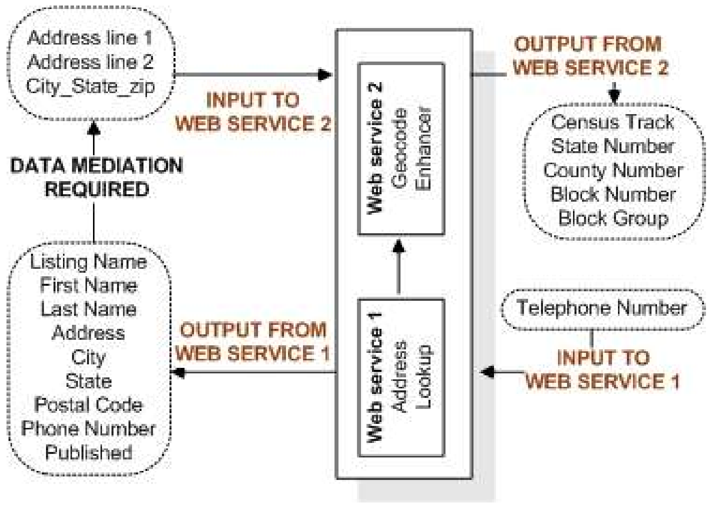


Figure 1 Process showing need for Data Mediation

3. Message-Level Heterogeneities

We define message or data level heterogeneities to exist between interoperating Web services when the data elements that have to be passed between the two services are incompatible. Although SOAP (XML-based messaging) allows message exchange between services with heterogeneous message formats, the data itself is rendered useless or incorrect by the Web service receiving the message. Data mediation between the services, i.e., transforming one message format to another is required.

Our solution for data mediation borrows from the field of schema/data integration in federated databases. Conceptually, schema/data integration can be divided into two parts - schema matching and schema/data mapping. Finding semantic correspondences between elements of two schemas is called *Matching*. *Mapping* deals with the physical representation of the matches established by schema matching and rules for transforming elements of one schema to that of the other. In this paper, we focus on **data mediation in a Web services based environment using pre-defined**

mappings. A discussion on how the autonomous nature of Web services makes the problem of matching and mapping more challenging than in the database domain is presented in Section 10.

In both databases and Web services, automating the process of matching and mapping is hard due to heterogeneities at the following levels (Sheth, 1992), (Sheth, 1998):

- *Syntactic heterogeneity* - differences in the language used for representing the elements;
- *Structural heterogeneity* - differences in the types, structures of the elements;
- *Model/Representational heterogeneity* - differences in the underlying models (database, ontologies) or their representations (relational, object-oriented, RDF, OWL);
- *Semantic heterogeneity* - where the same real world entity is represented using different terms (or structures) or vice versa.

Previous work on classifying schematic heterogeneities in databases (Kim, 1993), (Kashyap, 1996) included heterogeneities at all four levels. In the context of Web services, syntactic and model/representational heterogeneities between service message elements are not relevant since the XML based environment automatically resolves them. Adapting from previous work, we classify structural and semantic message level heterogeneities as:
(a) **Attribute level incompatibilities** that arise when semantically similar *attributes* are modeled using different descriptions. These include Naming, Data Representation and Data Scaling conflicts.
(b) **Entity definition incompatibilities** that arise when semantically similar *entities* are modeled using different descriptions. These include Naming and Schema Isomorphism conflicts.
(c) **Abstraction level incompatibilities** that arise when two semantically similar *entities* or *attributes* are represented at different levels of abstraction. These include Generalization, Aggregation and Attribute Entity conflicts.

Section 3.1 and Table 1 present each of these conflicts and suggest how one might resolve them using semantic annotations and/or mappings between the services. In Section 3.2, we illustrate as many of these conflicts that exist between two interoperable real-world Web services.

3.1 Classifying Message-Level Heterogeneities

One of the hardest challenges in automating the process of generating transformations or mappings between service messages to facilitate message-level interoperability is the varied conflicts that exist between the messages exchanged. Adapting from past work in databases (Kim, 1993), (Kashyap, 1996), we classify service message-level structural and semantic heterogeneities (also refer to Table 1) and examine the challenges in automating the process of creating these mappings. Semantic heterogeneities hinder the process of identifying if two message schemas are semantically similar while structural heterogeneities present challenges in the automatic creation of the mappings between the message formats.

Attribute level Incompatibilities:

In this section we discuss attribute level incompatibilities when two different descriptors are used for semantically similar attribute domains. Consider two message schemas below:

STUDENT (Id#, Name)
STUDENT (SSN, Name)

where STUDENT.SSN is defined as a 9 digit number while STUDENT.Id# is defined as a 4 digit number (possibly the last 4 digits of the SSN).
There are two conflicts that are apparent here:

1. a semantic *naming conflict*, where two attributes that mean the same thing i.e. are identifiers for a student are called different things (Id# and SSN)
2. a structural or syntactic *data representation conflict* where Id# and SSN are defined as 4 digit and 9 digit numbers respectively.

Once the semantic conflict has been resolved, creating the mapping to go from the SSN to the Id# can possibly be automated while going in the reverse direction, i.e., transforming the Id# to a SSN requires additional information. Challenges of such kinds make the automatic generation of mappings almost impossible.

Another class of conflicts in this space is the *data scaling conflict* as shown in Table 1. Two attributes Marks (1-100) and Grades (A-F) that are semantically similar, but not compatible need additional information or human involvement before they can be mapped to each other.

Table 1 Message Level Heterogeneities

Heterogeneities / Conflicts	Examples - conflicted elements shown in color	Suggestions / Issues in Resolving Heterogeneities
Attribute level Incompatibilities – *differences that arise because of using different descriptions for semantically similar attributes*		
Naming conflicts Two attributes that are semantically alike might have different names (synonyms) Two attributes that are semantically unrelated might have the same names (homonyms)	*Web service 1* Student(Id#, Name) *Web service 2* Student(SSN, Name) *Web service 1* Student(Id#, Name) *Web service 2* Book (Id#, Name)	A semantic annotation on the entities and attributes (provided by *SAWSDL:modelReference*) will indicate their semantic similarities.
Data representation conflicts Two attributes that are semantically similar might have different data types or representations	*Web service 1* Student(Id#, Name) Id# defined as a 4 digit number *Web service 2* Student(Id#, Name) Id# defined as a 9 digit number	* Mapping WS2 Id# to WS1 Id# is easy with some additional context information while mapping in the reverse direction is most likely not possible.
Data scaling conflicts Two attributes that are semantically similar might be represented using different precisions	*Web service 1* Marks 1-100 *Web service 2* Grades A-F	* Mapping WS1 Marks to WS1 Grades is easy with some additional context information while mapping in the reverse direction is most likely not possible.
Entity level Incompatibilities – *differences that arise because of using different descriptions for semantically similar entities*		
Naming conflicts Semantically alike entities might have different names (synonyms) Semantically unrelated entities might have the same names (homonyms)	*Web service 1* EMPLOYEE (Id#, Name) *Web service 2* WORKER (Id#, Name) *Web service 1* TICKET (TicketNo, MovieName) *Web service 2* TICKET(FlightNo, Arr. Airport, Dep. Airport)	A semantic annotation on the entities and attributes (provided by *SAWSDL:modelReference*) will indicate their semantic similarities.
Schema Isomorphism conflicts Semantically similar entities may have different number of attributes	*Web service 1* PERSON (Name, Address, HomePhone, WorkPhone) *Web service 2* PERSON (Name, Address, Phone)	* Mapping in both directions will require some additional context information.
Abstraction Level Incompatibility – *Entity and attribute level differences that arise because two semantically similar entities or attributes are represented at different levels of abstraction*		
Generalization conflicts Semantically similar entities are represented at different levels of generalization in two Web services	*Web service 1* GRAD-STUDENT (ID, Name, Major) *Web service 2* STUDENT(ID, Name, Major, Type)	* WS2 defines the student entity at a much general level. A mapping from WS1 to WS2 requires adding a Type element with a default 'Graduate' value, while mapping in the other direction is a partial function.
Aggregation conflicts Two semantically similar entities, one represented as an aggregate of another	*Web service 1* PROFESSOR (ID, Name, Dept) *Web service 2* FACULTY (ID, ProfID, Dept)	* A set-of Professor entities is a Faculty entity. When the output of WS1 is a Professor entity, it is possible to identify the Faculty group it belongs to, but generating a mapping in the other direction is not possible.
Attribute Entity conflicts Semantically similar entity modeled as an attribute in one service and as an entity in the other	*Web service 1* COURSE (ID, Name, Semester) *Web service 2* DEPT(Course, Sem, .., ..)	* Course modeled as an entity by WS1 is modeled as an attribute by WS2. With definition contexts, mappings can be specified in both directions.

*** Interoperation between services needs transformation rules (mapping) in addition to annotation of the entities and/or attributes indicating their semantic similarity (matching).**

Entity level Incompatibilities:

Similar conflicts can occur at the entity level where different descriptors are used to describe semantically similar entities. Table 1 illustrates two conflicts in this class:

1. a *naming conflict* where different words are used to describe an EMPLOYEE or WORKER entity while the word TICKET is used to represent two semantically incompatible entities.
2. a *schema isomorphism conflict* where semantically similar PERSON entities are schematically or structurally incompatible because of different number and/or type of attributes. Once a semantic resolution is achieved between the PERSON entities, it is not always possible to

automate the generation of mappings when there is not enough information available (i.e. missing attributes in one schema).

Entity and Attribute level Abstraction Incompatibilities:

Some entity and attribute level conflicts occur because of how they are modeled at different levels of abstraction. Table 1 shows three classes of such conflicts:

1. a *generalization conflict* between two entities or attributes may occur if they are represented at different levels of generalizations in two schemas. For example, two services could model semantically similar student entities either as a GRAD-STUDENT or a STUDENT in two schemas, where the latter is defined more generally than the former. Generating mappings in both directions call for different requirements – the mapping from a STUDENT to a GRAD-STUDENT is a partial function (not every student is a graduate student) while in the other direction, the appropriate attribute value for Type has to be added in the mapping.
2. an *aggregation conflict* when aggregation is used in one schema to identify a group of entities in the other schema. For example, a PROFESSOR and FACULTY entity as defined in Table 1. Given a PROFESSOR entity, it is possible to create a mapping using additional information to map to the FACULTY entity; but generating a mapping in the reverse direction is not possible.
3. an *attribute entity conflict* where a concept is modeled as an attribute in one schema and as an entity in another. For example, the concept COURSE in the following schemas is modeled at different abstractions:
   COURSE (Id, Name, Semester)
   DEPT (Course, Semester …)
   Generating mappings in both directions will require some amount of human input in restructuring the schemas and obtaining additional contextual information, although automating them is hard.

3.2 Message-Level Heterogeneities - Examples from real-world Web Services

In this section, we examine the two interoperable real world Web services presented in the motivation scenario in section 2; identify the different types of message level conflicts and relate them to the classification we presented in the earlier section. The first Web service is a directory listing Address Lookup service (available at (PhoneLookupService)) that returns an address for a listed telephone number. The second Web service (available at (GeocodeService)) is a Geocode Enhancer service that uses an address to provide demographic and logistical information. Both these services put together as a process for a marketing analyst is shown in Figure 1. Seamless interoperability between these two services is hindered by the conflicts between the output and input message formats of the services. The message formats from both the Web services and the conflicts between them are shown in Figure 2.

At the attribute level, the 'Address' attribute in the 'Listing' complex type and the 'addressLine1' and 'addressLine2' attributes in the 'GetGeoCodeUSA' are semantically equivalent entities, but are called different things and also modeled differently. A mapping between the two would entail a relatively simple concatenation between the ‘addressLine1’ and ‘addressLine2’ attributes to map to the 'Address' attribute. A similar explanation holds for the attributes 'City', 'State' and 'PostalCode' in the 'Listing' complex type vs. the 'city_state-zip' attribute in the 'GetGeoCodeUSA' complex type.

At the entity level, it is fairly obvious to see that the two entities 'Listing' and 'GetGeoCodeUSA' are modeled very differently mostly because the services themselves were meant for different purposes. In reusing the service functionalities, the messages present schema isomorphism and naming conflicts.

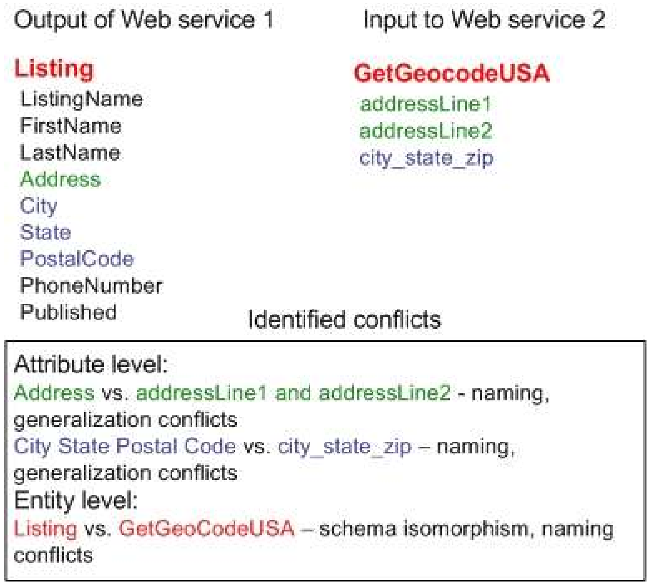


Figure 2 Illustrating message-level heterogeneities

Our approach for resolving these message-level conflicts and achieving interoperability between services is based on the data mediation support provided by WSDL-S (WSDL-S). Now as a part of a chartered working group for semantic Web services SAWSDL, this approach of data mediation is showing good promises of adoption.

4. Evolutionary Approach to Semantic Web Services: WSDL-S and SAWSDL

SAWSDL (SAWSDL) provides a mechanism to annotate the capabilities and requirements of Web services (described using WSDL) with semantic concepts defined in an external domain model. SAWSDL is based on an earlier work WSDL-S (WSDL-S), a W3C member submission for Semantic Web services. Using XML extensibility elements and attributes, semantic annotations on WSDL elements (including inputs, outputs and functional aspects like operations, their preconditions and effects) are achieved by referencing semantic concepts from one or more external domain models (ontology). Externalizing the domain models allows WSDL-S to take an agnostic view towards semantic representation languages. This allows developers to build domain models in any preferred language or reuse existing domain models. This is an advantage, since before OWL was popular, quite a few domain models were developed using RDF/S (RDF) and

UML . Figure 3 shows a pictorial representation of how semantics are associated with WSDL elements by referencing concepts in external semantic domain models.

Of the extensibility attributes on interfaces, operations, faults and message elements; the *modelReference* and *schemaMapping* extensibility attributes on message elements are the most relevant to this work. A summary of these extension attributes defined by SAWSDL is given below:

- the *modelReference* attribute is used to specify the association between a WSDL element and a concept in some semantic model. It can be used to annotate XSD complex type definitions, simple type definitions, element declarations, attribute declarations as well as WSDL interfaces, operations, and faults.
- two extension attributes *liftingSchemaMapping* and *loweringSchemaMapping*, that are added to XML Schema element declarations, complex type definitions and simple type definitions for specifying mappings between the schema elements and their corresponding semantic model concepts.

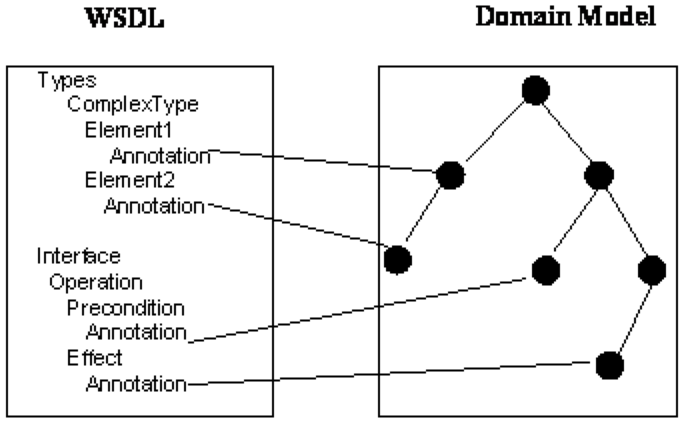


Figure 3 Externalized representation and association of semantics to WSDL elements

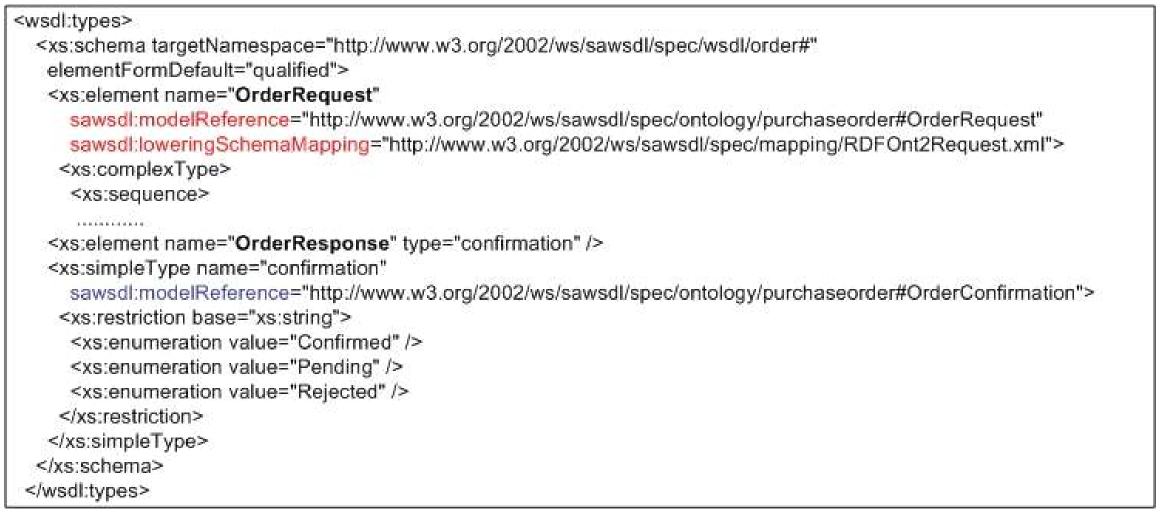
The *modelReference* attribute on an input or output message element indicates a semantic similarity between the message element and a concept in the Ontology. It is possible that beyond this semantic match, there is a structural conflict between the XML message element and Ontology concept. In such cases, the *liftingSchemaMapping* extension attribute is used to specify a mapping or transformation going from the XML element to the Ontology concept. The *loweringSchemaMapping* is used to specify the mapping in the reverse direction, i.e., from the Ontology concept to the XML element.

Model references are used in determining if a service meets the requirements of a client (during discovery and composition) and schema mappings resolve any further mismatches in the structure of the inputs and outputs (during actual invocation) of the services. Sections 5 and 6 illustrate how our approach to data mediation uses these mappings to interoperate seamlessly between services.

In the following sections we show how SAWSDL associates semantics and mappings with WSDL elements, illustrate with some examples, and follow it with a discussion on representing the mappings associated with the *schemaMapping* extension attributes.

4.1 SAWSDL Example

The current SAWSDL working draft (SAWSDL) illustrates the extension attributes using a purchase order Web service. This fictitious Web service expects as input a customer account number and a list of items to be ordered, containing quantity information and a product identifier in form of a Universal Product Code (UPC). The service returns as an output the status of the order, which can be either reject, accept or pending. Here, we discuss in detail only the annotations on the message elements of the service. Figure 4 below, shows an excerpt of a sample SAWSDL file (from the SAWSDL draft) with extensibility attributes on the message elements 'OrderRequest 'and 'OrderResponse'.



```
<wsdl:types>
  <xs:schema targetNamespace="http://www.w3.org/2002/ws/sawsdl/spec/wsdl/order#"
   elementFormDefault="qualified">
   <xs:element name="OrderRequest"
     sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#OrderRequest"
     sawsdl:loweringSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapping/RDFOnt2Request.xml">
    <xs:complexType>
     <xs:sequence>
      ............
   <xs:element name="OrderResponse" type="confirmation" />
   <xs:simpleType name="confirmation"
     sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder#OrderConfirmation">
    <xs:restriction base="xs:string">
     <xs:enumeration value="Confirmed" />
     <xs:enumeration value="Pending" />
     <xs:enumeration value="Rejected" />
    </xs:restriction>
   </xs:simpleType>
  </xs:schema>
 </wsdl:types>
```

Figure 4 Excerpt of a sample SAWSDL file

The *modelReference* annotation on the 'OrderRequest' message element points to the OrderRequest concept in a purchaseOrder Ontology. In addition to the *loweringSchemaMapping* in this example, the specification also shows a *liftingSchemaMapping* example in Appendix A.

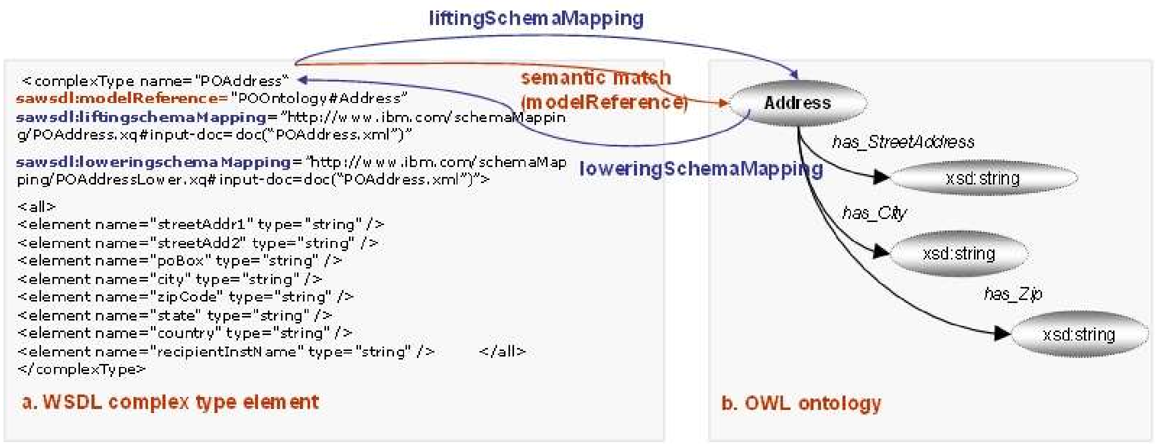
Given the complexity of the messages (purchase orders have many fields) and the domain model concepts used in this example, we illustrate the idea behind specifying mappings i.e. the *liftingSchemaMapping* and *loweringSchemaMapping,* with the help of a simpler example below. These mappings are central to our data mediation approach and warrant a detail explanation.

SAWSDL Schema Mappings

Consider the WSDL message element and Ontology concept as shown in Figure 5a and 5b. The 'Address' complex type in the WSDL and the 'Address' concept in the OWL Ontology are semantically equivalent entities (indicated by the *modelReference* attribute), but are not structurally equivalent.

An excerpt of the *liftingSchemaMapping* attribute on the WSDL 'Address' element is shown in a. The mapping transforms an instance of the WSDL XML element to an instance of the Ontology concept. The structural conflict in this case was a *generalization conflict* and relatively easy to resolve by using a concatenation function. The SAWSDL specification itself does not prescribe a specific mapping language, however, for the purpose of illustration, a set of concrete technologies is chosen. To illustrate various mapping representation options, the examples in the specification use XSLT and SPARQL for representing mappings, while the mappings shown in are represented using XQuery constructs.

An excerpt of the *loweringSchemaMapping* attribute on the WSDL 'Address' element is shown in b. A *loweringSchemaMapping* is used to transform a semantic concept to a WSDL XML element instance. In this case, it entails splitting the value of the attribute 'has_StreetAddress' into two simple WSDL elements 'streetAddress1' and 'streetAddress2'. Automatically generating this mapping is hard since it might require some human input to determine where the address string needs to be split (in this example delimited by a comma).



```
 <complexType name="POAddress"
sawsdl:modelReference="POOntology#Address"
sawsdl:liftingschemaMapping="http://www.ibm.com/schemaMappin
g/POAddress.xq#input-doc=doc("POAddress.xml")"

sawsdl:loweringschemaMapping="http://www.ibm.com/schemaMap
ping/POAddressLower.xq#input-doc=doc("POAddress.xml")">

<all>
<element name="streetAddr1" type="string" />
<element name="streetAdd2" type="string" />
<element name="poBox" type="string" />
<element name="city" type="string" />
<element name="zipCode" type="string" />
<element name="state" type="string" />
<element name="country" type="string" />
<element name="recipientInstName" type="string" />        </all>
</complexType>
```

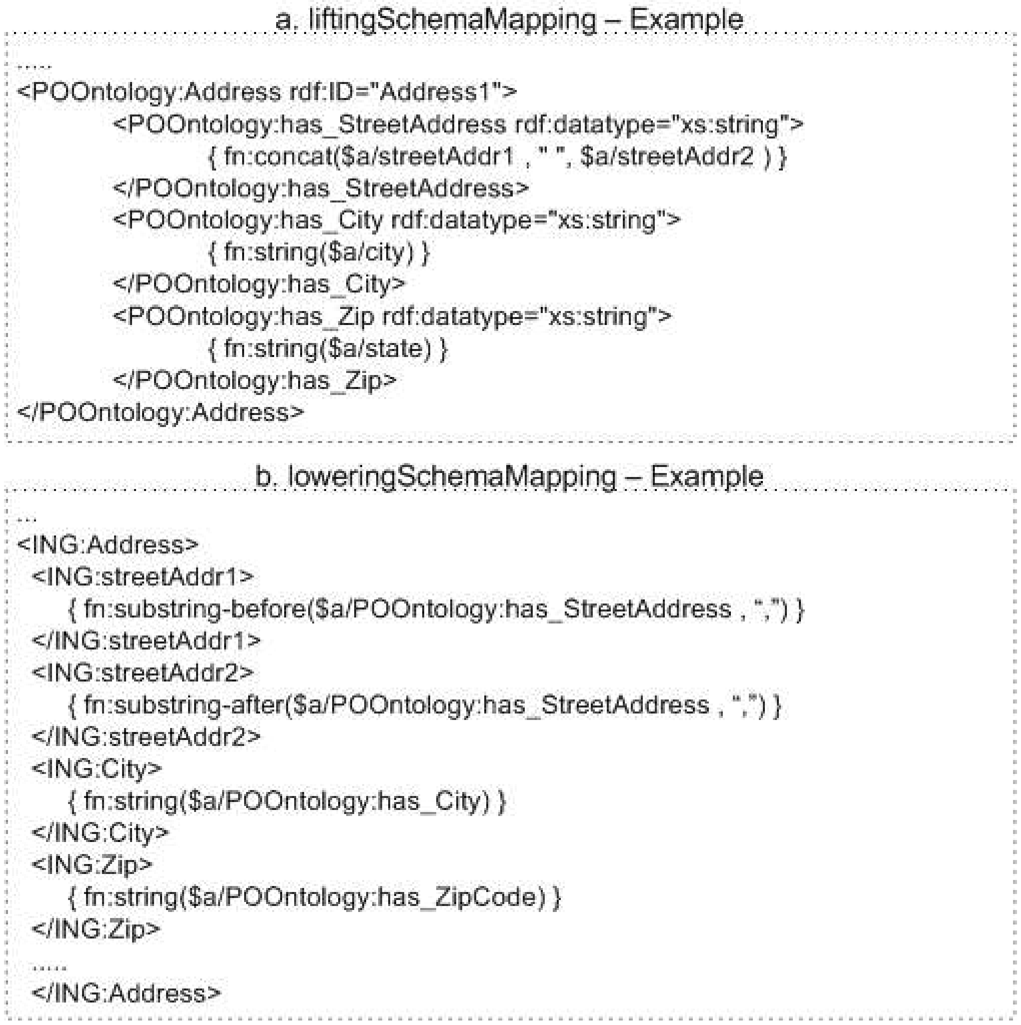
Figure 5 SAWSDL schema mappings - Example

The schema definitions of the *liftingSchemaMapping* and *loweringSchemaMapping* attributes can be found in the SAWSDL specification.

4.2 Representing Mappings

In addition to the actual process of matching and generating mappings, the representation of the mappings is also of significant concern. The expressiveness of the mapping language can dictate to a large extent, the types of heterogeneities that can be resolved. Some of the past approaches to representing mappings have been *queries or views* (D. Calvanese, 2001), XQuery, XSLT; *mapping tables* (Kementsietsidis A. , 2003); *bridging axioms* (Madhavan, 2002), (Dou, 2003); as *instances in an ontology of mappings* (Maedche A 2002), (Crub´ezy, 2003); *languages* (S.B. Davidson, 1995), (P. Bouquet, 2003), (Kim, 1993), etc.

In this work, we use a popular representation for mappings in Web services, *XQuery and XSLT* , (Onose, 2004) and use SAWSDL to associate these mappings with Web service elements. We believe that most of the mappings that are required to resolve heterogeneities between Web service elements can be concisely represented using XQuery or XSLT. In some cases, when the lack of a canonical XML representation of RDF makes the use of XSLT ill suited for representing mappings, SPARQL can be used as an alternative. The developer however has the flexibility of using any mapping language since SAWSDL is agnostic to the mapping representation used. Since our implementation for data mediation exploits this feature of SAWSDL, it is also independent of the mapping or conceptual model representation used.



a. liftingSchemaMapping – Example

```
.....
<POOntology:Address rdf:ID="Address1">
        <POOntology:has_StreetAddress rdf:datatype="xs:string">
                { fn:concat($a/streetAddr1 , " ", $a/streetAddr2 ) }
        </POOntology:has_StreetAddress>
        <POOntology:has_City rdf:datatype="xs:string">
                { fn:string($a/city) }
        </POOntology:has_City>
        <POOntology:has_Zip rdf:datatype="xs:string">
                { fn:string($a/state) }
        </POOntology:has_Zip>
</POOntology:Address>
```

b. loweringSchemaMapping – Example

```
...
<ING:Address>
  <ING:streetAddr1>
     { fn:substring-before($a/POOntology:has_StreetAddress , “,”) }
  </ING:streetAddr1>
  <ING:streetAddr2>
     { fn:substring-after($a/POOntology:has_StreetAddress , “,”) }
  </ING:streetAddr2>
  <ING:City>
     { fn:string($a/POOntology:has_City) }
  </ING:City>
  <ING:Zip>
     { fn:string($a/POOntology:has_ZipCode) }
  </ING:Zip>
  .....
  </ING:Address>
```

Figure 6 Lifting and Lowering mappings - examples

4.3 Creating a SAWSDL file from a WSDL file

Radiant (Gomadam, 2005), (Radiant), (Rajasekaran, 2004) is an eclipse plug-in that provides a user interface for annotating existing WSDL documents using an OWL Ontology to create a SAWSDL file. Radiant comes with an ontology viewer based on the UMBC's Cobra Ontology Viewer . In addition to this feature, it also supports publishing of the SAWSDL file using the jUDDI , an open source Java implementation of the Universal Description, Discovery, and Integration (UDDI) specification for Web Services.

5. Proposed Data Mediation Approach

Support for data mediation in SAWSDL is provided by having the developer associate mappings (created either manually or using semi-automatic tools) using the *'schemaMapping'* attributes on Web service message (input and output) elements. Mappings are created between the Web service message element and the ontology concept with which the message element is semantically associated, as depicted in Figure 5 and . In addition to a mapping from the Web service message element to the ontology concept, also called the *liftingSchemaMapping*, an additional mapping from the ontology concept to the message element, called the *loweringSchemaMapping*, is also specified. Once the mappings are defined, two Web services can interoperate by reusing these mappings. The ontologies now become a vehicle through which Web services resolve their message level heterogeneities. For the sake of simplicity, the ontologies used are created using OWL, although SAWSDL is agnostic to the domain model representation language.

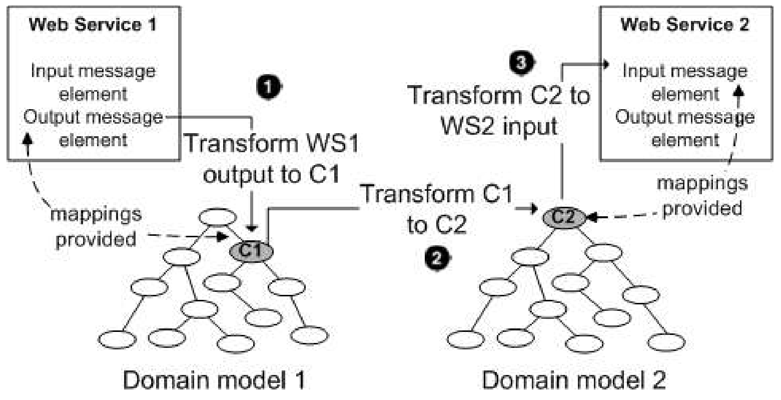


Figure 7 Data transformation using SAWSDL

Data transformation proceeds in three steps as shown in . In the first step (1), the output message of Web Service 1 (WS1) is transformed to the OWL concept to which it is mapped *(liftingSchemaMapping)*; next, the OWL concept is transformed to the input message of Web service 2 (WS2) (3) *(loweringSchemaMapping)*. In the event that mappings in the two Web services are not provided using the same ontology, mappings between the ontology concepts C1 and C2 are required to enable data mediation (2) (see Section 5.1 for a discussion on inter-ontology matching and mapping). Although the mappings are defined at the schema level between the WSDL (XML) and OWL schemas, the message transformation occurs at the instance level between the WSDL (XML) and OWL instance.

5.1. Inter Ontology Mappings - Discussion

As ontologies become popular, it is conceivable that mediation is needed between services that are mapped using two different conceptual models. (i.e., two services that need to interoperate are mapped using ontologies created from Rosetta Net PIPs (RosettaNet) and ebXML CCD (ebXML)) In such a case, there would be a need for inter-ontology mappings. Matching and mapping of ontology schemas is a vast area of research and has seen plenty of advances that can be used to address this concern. Among other work in this area, (Mena, 1996), (D. Calvanese, 2001) and (Kalfoglou, 2003) discuss this problem in different contexts and provide useful insights.

In the context of this work, when two Web services have specified lifting and lowering mappings to two different Ontologies, mappings between the Ontologies encoded using SPARQL or other mapping representation languages are also executed to transform instances of one Ontology to another. While we plan to extend our approach to handle multiple ontology matching and mapping, our current solution uses a single Ontology for data mediation. This is still is useful as number of popular specifications/ontologies (ebXML, RosettaNet, OAGIS, etc.) are currently being used for interoperability between business partners.

6. SYSTEM ARCHITECTURE

The general philosophy of the METEOR-S project, as outlined in (Sivashanmugam, 2003) and (Verma, 2006), has been to use the extensibility elements of Web service standards to add semantics to Web services. An important manifestation of this approach is of course, WSDL-S and therefore SAWSDL. A key motivator for us to follow this approach was the ease with which we could incorporate tooling support for Semantic Web services in existing tools. The system architecture in this paper is a validation of our philosophical choice, as we use the extensibility support provided by Axis 2 (specifically the ability to add user modules) to propose a solution for data mediation.

The system architecture shown in Figure 8 consists of a main METEOR-S middleware component implemented as modules on an Axis2 server. The actual METEOR-S middleware component comprises of several modules (which in turn consist of handlers) for achieving functionalities like semantic Web service publishing, discovery, composition, etc. In the interest of space and clarity, the system architecture illustrates only an End Point Resolution (EPR) handler, a Data Mediation (DM) handler and their functionalities. In this section, we will describe the two handlers and illustrate how data mediation is achieved in a process akin to the one in Figure 1.

Before using the METEOR-S data mediation functionality, the only tasks the developers are required to perform are the following:

- Web services should be described using SAWSDL by annotating the WSDL file with semantic concepts from an ontology (using a tool like (Gomadam, 2005)). The *liftingSchemaMapping* and *loweringSchemaMapping* from the Web service message elements to the semantic concepts should be created.
- The Web services must be deployed and the WSDL-S files must be accessible. Axis 2 allows deployment of WSDL-S files.

EPR Handler: The METEOR-S middleware may reside at any machine. In order for Web service clients to take advantage of the provided data mediation support, their SOAP messages must be routed to the METEOR-S middleware. This is done using a small client side utility that has two functionalities:
Change the EPR of the Web service being invoked to point to the METEOR-S middleware. This new EPR is called the *logical EPR*.
Contact the METEOR-S middleware to register a mapping of the *logical EPR* to the actual EPR of the Web service.
The End Point Resolution *(EPR)* handler is responsible for changing the incoming SOAP message by replacing the *logical EPR* with the actual/physical EPR of the service. This allows the appropriate Axis handlers to redirect the message to the Web service after the data mediation handler has transformed the message.

DM Handler: The *DM* handler which is the main component for facilitating data interoperation works in cooperation with the *EPR* handler and a mapping processing engine to enable data mediation. Each time a Web service is invoked, the *DM* handler obtains the *'schemaMapping'* functions from the Web service WSDL-S locations (using the WSDL-S4J API), performs the *lifting* and *lowering* mappings on the incoming SOAP message using a mapping processor/engine (SAXON for XQuery and XSLT) and then updates the SOAP message. Appropriate Axis handlers then invoke the Web service with the transformed message. SAXON (SAXON) is an open source XQuery/XSLT processor that we use to process the mappings represented using XQuery/XSLT.

6.1 Walk-through Example

In this section, we describe data mediation in a process with two Web services, where data mediation is required between the first and the second Web service. Our implementation is agnostic to how a process is represented. The evaluations were conducted using BPEL processes (BPEL); although users can emulate a process by chaining invocation of services in Java. Both Web services are described using SAWSDL and provide the necessary *'schemaMapping'* functions required to perform data mediation. For the sake of simplicity, let us assume that both the Web services have been annotated using the same ontology that has been created using OWL.

In Figure 8, steps 1a through 1e show the SOAP messages during the invocation of Web service 1 and steps 2a through 2e show SOAP messages when Web service 2 is being invoked. Both steps 1 and 2 show the use of the *EPR* handler and the *DM* handler.

Let us now walk through the figure to understand how data mediation is achieved.

Steps 1(2) a though 1(2) e: Invoking Web service 1(2)

Every time the client process invokes a partner Web service, the SOAP message is routed to the METEOR-S middleware because of the *logical EPR* setting in the Web service. In this section, we will trace the SOAP messages as they are processed by the middleware:

Step 1(2) a: The client generated SOAP message for invoking Web service 1(2) (shown as SOAP A in the figure) is now directed to the middleware server and passes through Axis 2.

Step 1(2) b: The EPR handler changes the SOAP message by replacing the *logical EPR* with the physical EPR of Web service 1(2). The new SOAP message is shown as SOAP B in the figure.

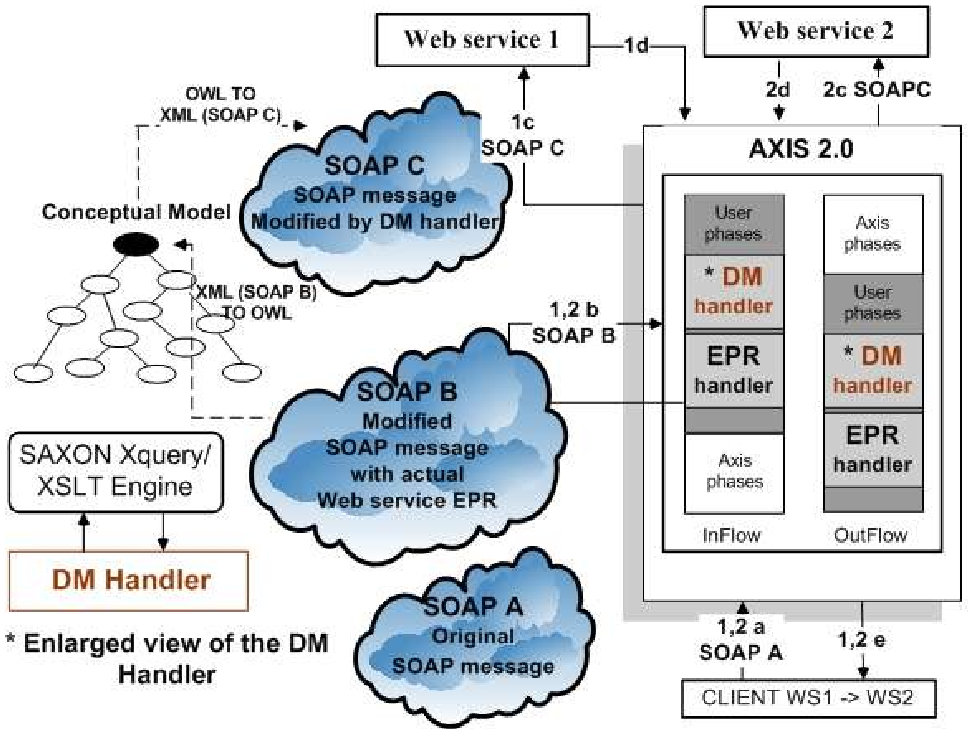


Figure 8 Data Mediation System Architecture

Step 1(2) c: Step 1c might be optional depending on the client's message to Web service 1. In this step, we will elucidate step 2c, when the output of Web service 1 needs to be transformed to the input of Web service 2. The *DM* handler uses the SAXON processor to convert the message intended for Web service 2 via the following steps (also see):

i. Using the namespaces in the SOAP message and the logical to physical map in the *EPR* module, the SAWSDL file is accessed to get the *'liftingSchemaMapping'* provided on the

output message element of the Web service (Web service 1) whose output is supplied to the Web service being invoked (Web service 2).
ii. Using the actual EPR of the Web service to be invoked, the SAWSDL of the actual Web service is accessed to get the *'loweringSchemaMapping'* mapping provided on the input message element of the Web service being invoked (Web service 2).
iii. The *'liftingSchemaMapping'* mapping that converts an XML message instance to an OWL instance, is used by SAXON to convert the message obtained from Web service 1 (SOAP B body content) to the OWL concept to which it is mapped (enlarged view of DM handler)
iv. The *'loweringSchemaMapping'* that converts the OWL instance to an XML message instance, is used by SAXON to convert the OWL object to an XML message (SOAP C) of the format that can be used by the Web service being invoked (enlarged view of DM handler)
v. The original content in the SOAP body, which was the message returned by the previously invoked Web service (Web service 1), is then replaced with the transformed XML message (shown as SOAPC in the figure). The transformed SOAP message (SOAP C) is forwarded to the actual Web service.

Step 1(2) d,e: The service replies back to the METEOR-S middleware that sends the message back to the client.

7. Evaluation

In an attempt to evaluate how many real world services today are perfectly interoperable, we created an 'investment assistant' process with an in-house service and tried to plug 'n' play real-world Web services. Using two external services that returned real time stock quotes and company profile information using a ticker symbol input, we built a process that takes the output of these services, additional user information on investing in this stock and returns the likelihood of a success on such an investment. The real-world Web services in the process are shown in grey boxes. The 'investment helper' service was created by an internal expert familiar with the finance domain but with no knowledge of the existing real-world Web service message schemas.

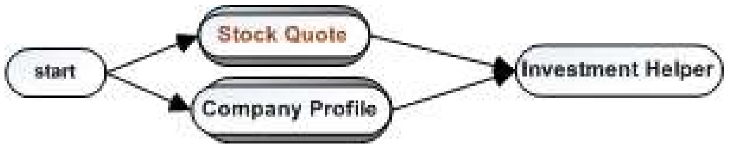


Figure 9 Investment Assistance Process

With this process in place, we tried to plug in real-world 'stock quote' Web services and evaluate how many would work without the need for any data mediation. The registries we used for discovering these services are popular, commonly used public registries listed in (UDDIregistries). Table 2 shows the statistics of this evaluation. Of the ten semantically relevant services that we found, none could interoperate with the 'investment helper' service without the use of data mappings. Three of the ten services could interoperate with the use of simple mappings, while one could not interoperate at all because of insufficient information in the message. The reader should notice that irrespective of the message schema of the 'investment helper' service, a majority of services would need support for data interoperability.

Table 2 Web service interoperability - Evaluation

URI of stock quote Web services	Message-level Heterogeneities					Can achieve interoperability using mappings
	Structural	Schema Isomorphism	Attribute Naming	Entity Naming	Data Repres.	
http://ws.strikeiron.com/SwanandMokashi/StockQuotes?wsdl	Yes	Yes	Yes	Yes	Yes	Yes
* http://ws.strikeiron.com/HistoricalStockQuotes?wsdl	Yes (minor)	Yes	No	Yes	No	Yes
http://ws.strikeiron.com/BasicRealTimeQuotes?wsdl	Yes	Yes	Yes	Yes	Yes	Yes
* http://www.webservicex.net/stockquote.asmx?wsdl	Yes (minor)	Yes	Yes	Yes	No	Yes
http://www.xmethods.net/sd/StockQuoteService.wsdl	Yes	Yes	Yes	Yes	Yes	Yes
glkev.webs.innerhost.com/glkev_ws/StockServices.asmx?wsdl	Yes	Yes	Yes	Yes	Yes	Yes
www.gama-system.com/webservices/stockquotes.asmx?wsdl	Yes	Yes	Yes	Yes	No	No
glkev.webs.innerhost.com/glkev_ws/HistoricalStockQuotes.asmx?wsdl	Yes	Yes	Yes	Yes	Yes	Yes
ws.cdyne.com/delayedstockquote/delayedstockquote.asmx?wsdl	Yes	Yes	Yes	Yes	Yes	Yes
* www.xignite.com/xquotes.asmx?WSDL	Yes (minor)	Yes	Yes	Yes	No	Yes

All stock quote Web services to interoperate with investment helper service – available at http://lsdis.cs.uga.edu/~meena/ICWS06/Eval.html

*** These services could interoperate with the investment helper service using very minor mappings between the message schemas**

This simple evaluation shows the importance of data mediation in ensuring interoperability of services. For each of the incompatible message formats, we were able to define mappings to a finance ontology (adapted from the finance domain of the SUMO Ontology (SUMO) and available at (Evaluations) using XQuery and use the proposed data mediation approach to interoperate between the services. In the interest of space, we have not shown the mapping expressions; the list of services used and the XQuery mappings are available at (Evaluations).

As we can see, an important benefit of this data mediation approach is the ability to plug 'n' play real world Web services in a process. The advantage is in the number of mappings one has to specify in this approach vs. when mappings have to be specified between services. The next section describes our experiences with dynamic processes that are characterized by the need to replace partners at run time and how the use of SAWSDL and this data mediation approach makes this feasible.

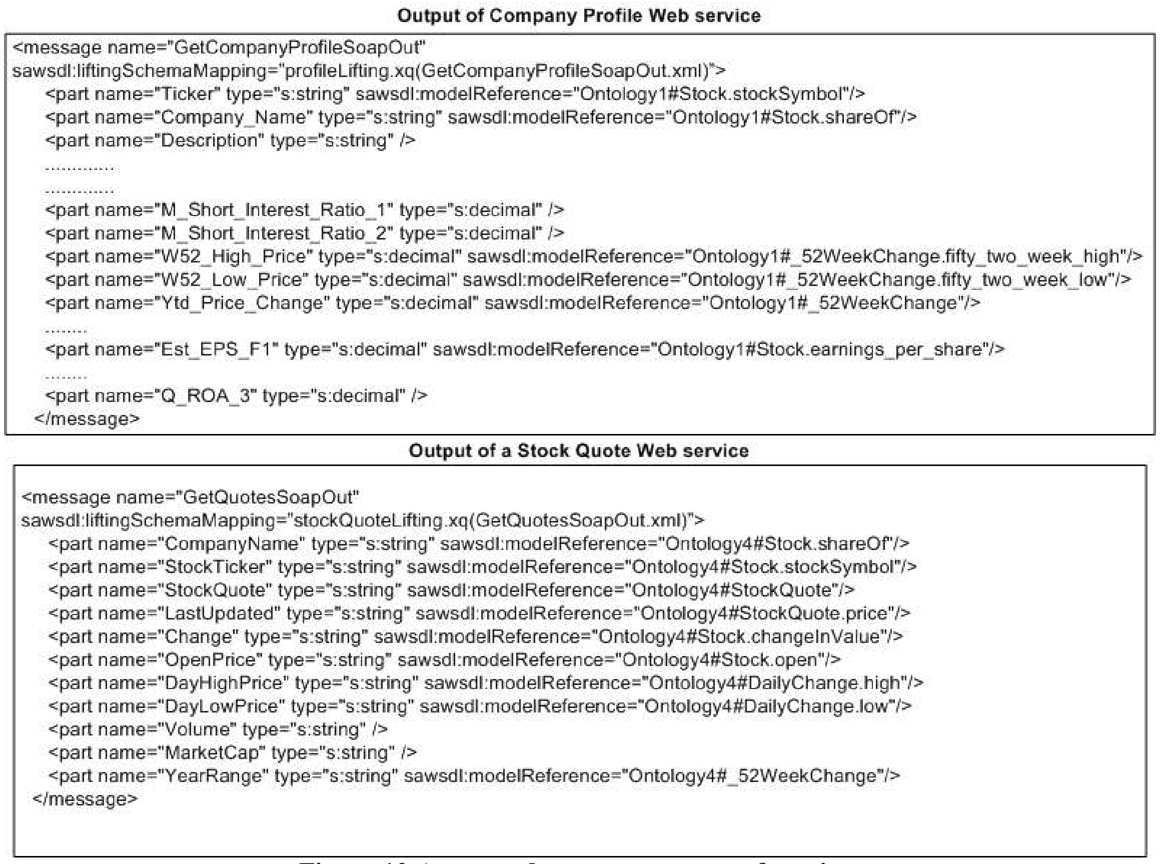
8. Runtime binding of Partner Services in a Web Process

Over the last couple of decades, workflow technology has increasingly been used to coordinate activities in inter and intra organizational settings. The large scale standardization of all aspects of businesses has set the stage for businesses to configure their processes on the fly with new or pre-existing business partners. Such a run time reconfiguration of processes requires not only a semantic functional agreement between the services, but also perfect data interoperability. In this section, we illustrate how business processes can be re-configured on the fly if their partner services are annotated using SAWSDL.

Using the same process as in the Evaluation scenario, we show how the use of SAWSDL can allow not only the identification of alternate partner services for a process, but also facilitate a runtime binding of the services. As mentioned earlier, all service WSDL files are available online from the sources listed in Table 2. The SAWSDL files of all the stock quote services, the company profile service (obtained from www.strikeiron.com) and the in-house investment helper service are all available for download at (Evaluations).

Both the stock quote and the company profile services (see Figure 9) take a stock ticker as an input and their outputs are collectively used as an input for the investment helper service. Figure 10 shows the annotated output messages of the stock quote and company profile services. Since the output of the company profile service was very big, we have shown only an excerpt in the figure. shoes the annotated input message that the investment helper service requires. For the sake of simplicity all services were annotated using the same Ontology (http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/ontologies/LSDIS_Finance.owl) and using the Radiant tool. The last two message parts in , quantity and transaction amount are user input parameters to the investment helper service.

As one can see, the output messages and the input message are heterogeneous with different parts of the output messages mapping to the input message parts. The message structures are relatively simple but have several naming and schema isomorphism conflicts between them. Passing the collective outputs of the services as they are will result in an error while invoking the investment helper service because the output instance passed will not satisfy the schema definition of the input message. A mapping is required to map the relevant output parts to the relevant input parts. The *modelreference* annotations on the message help in identifying what the semantically similar entities are; which is further used in defining the lifting and lowering *schemaMappings* from the WSDL elements to the Ontology concepts.

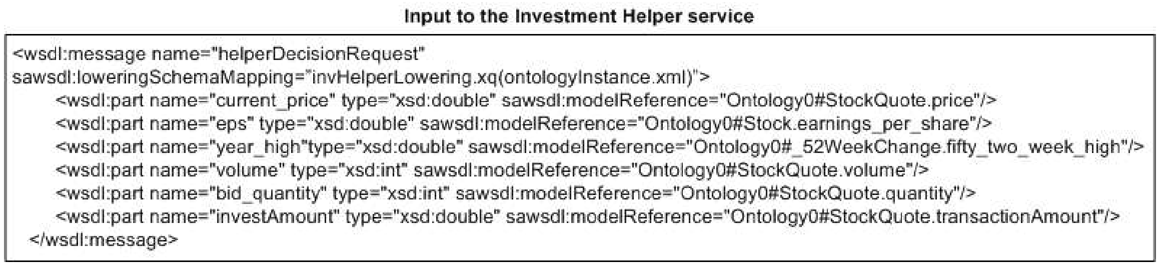

Output of Company Profile Web service

```
<message name="GetCompanyProfileSoapOut"
sawsdl:liftingSchemaMapping="profileLifting.xq(GetCompanyProfileSoapOut.xml)">
    <part name="Ticker" type="s:string" sawsdl:modelReference="Ontology1#Stock.stockSymbol"/>
    <part name="Company_Name" type="s:string" sawsdl:modelReference="Ontology1#Stock.shareOf"/>
    <part name="Description" type="s:string" />
    .............
    .............
    <part name="M_Short_Interest_Ratio_1" type="s:decimal" />
    <part name="M_Short_Interest_Ratio_2" type="s:decimal" />
    <part name="W52_High_Price" type="s:decimal" sawsdl:modelReference="Ontology1#_52WeekChange.fifty_two_week_high"/>
    <part name="W52_Low_Price" type="s:decimal" sawsdl:modelReference="Ontology1#_52WeekChange.fifty_two_week_low"/>
    <part name="Ytd_Price_Change" type="s:decimal" sawsdl:modelReference="Ontology1#_52WeekChange"/>
    ........
    <part name="Est_EPS_F1" type="s:decimal" sawsdl:modelReference="Ontology1#Stock.earnings_per_share"/>
    ........
    <part name="Q_ROA_3" type="s:decimal" />
  </message>
```

Output of a Stock Quote Web service

```
<message name="GetQuotesSoapOut"
sawsdl:liftingSchemaMapping="stockQuoteLifting.xq(GetQuotesSoapOut.xml)">
    <part name="CompanyName" type="s:string" sawsdl:modelReference="Ontology4#Stock.shareOf"/>
    <part name="StockTicker" type="s:string" sawsdl:modelReference="Ontology4#Stock.stockSymbol"/>
    <part name="StockQuote" type="s:string" sawsdl:modelReference="Ontology4#StockQuote"/>
    <part name="LastUpdated" type="s:string" sawsdl:modelReference="Ontology4#StockQuote.price"/>
    <part name="Change" type="s:string" sawsdl:modelReference="Ontology4#Stock.changeInValue"/>
    <part name="OpenPrice" type="s:string" sawsdl:modelReference="Ontology4#Stock.open"/>
    <part name="DayHighPrice" type="s:string" sawsdl:modelReference="Ontology4#DailyChange.high"/>
    <part name="DayLowPrice" type="s:string" sawsdl:modelReference="Ontology4#DailyChange.low"/>
    <part name="Volume" type="s:string" />
    <part name="MarketCap" type="s:string" />
    <part name="YearRange" type="s:string" sawsdl:modelReference="Ontology4#_52WeekChange"/>
  </message>
```

Figure 10 Annotated output messages of services



Input to the Investment Helper service

```
<wsdl:message name="helperDecisionRequest"
sawsdl:loweringSchemaMapping="invHelperLowering.xq(ontologyInstance.xml)">
      <wsdl:part name="current_price" type="xsd:double" sawsdl:modelReference="Ontology0#StockQuote.price"/>
      <wsdl:part name="eps" type="xsd:double" sawsdl:modelReference="Ontology0#Stock.earnings_per_share"/>
      <wsdl:part name="year_high"type="xsd:double" sawsdl:modelReference="Ontology0#_52WeekChange.fifty_two_week_high"/>
      <wsdl:part name="volume" type="xsd:int" sawsdl:modelReference="Ontology0#StockQuote.volume"/>
      <wsdl:part name="bid_quantity" type="xsd:int" sawsdl:modelReference="Ontology0#StockQuote.quantity"/>
      <wsdl:part name="investAmount" type="xsd:double" sawsdl:modelReference="Ontology0#StockQuote.transactionAmount"/>
  </wsdl:message>
```

Figure 11 Annotated input message to the Investment Helper service

Defining SAWSDL schema mappings

In line with our discussions in earlier sections, lifting and lowering mappings are specified from the WSDL message elements to the relevant semantic model concepts and in the other direction respectively. Here, we will illustrate only the mappings used in the interoperation, i.e.,

- The lifting mappings from the output message of the company profile service 'GetCompanyProfileSoapOut' to the relevant Ontology concept.
- The lifting mappings from the output message of the stock quote service 'GetQuotesSoapOut' to the relevant Ontology concepts.
- The lowering mappings from the Ontology concepts to the input message of the investment helper service ' helperDecisionRequest'

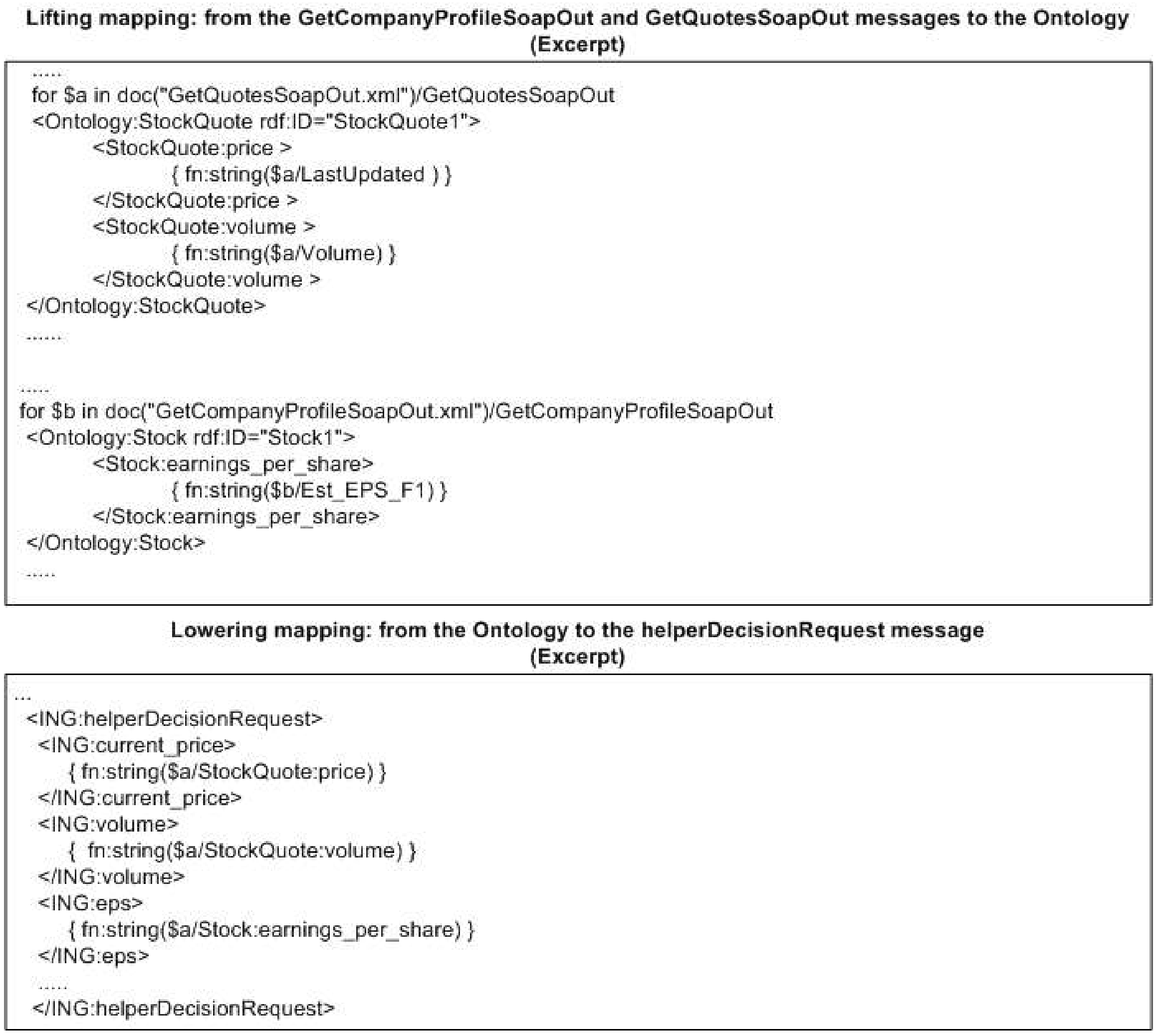
Figure 12 shows the lifting and lowering mappings in each case; the execution of which transforms output message instances to an OWL instance and then back to the input message instance for the investment helper service.

In creating these mappings, we chanced upon an interesting observation. Typically, semantic similarity is identified between objects and when needed, mappings are created between these objects. In this example, both the input and output messages are partially mapped to more than one Ontology concept. In other words, some parts of the messages map to parts of one Ontology class whereas some other parts map to a parts of a different Ontology class. Mappings in such cases are no longer simple object to object translations but a composite of multiple mapping constructs. The lifting mapping in Figure 12 illustrates one such case.

Runtime binding of partner services

In the absence of SAWSDL mappings, when services have to be replaced in a process, developers have to specify mappings between the new service and the existing services. The type of conflicts that might arise between the old and new services might be very complex as well, implying more time and domain expertise required to create those mappings. Appendix A shows examples of output messages of two other stock services in Table 2 to give readers an idea of the kind of message conflicts that exist between the interoperating services.

Previous work in this space in the context of dynamic binding services assumes a homogeneous nature of the original and new service messages (Mandell, 2003) (Sivashanmugam, 2004a). The inaccuracy of this assumption is evident in the examples of services we have presented in this section. Our work alleviates this assumption by having the services specify mappings to a semantic domain model and reusing the mappings for interoperation after runtime binding.


Lifting mapping: from the GetCompanyProfileSoapOut and GetQuotesSoapOut messages to the Ontology (Excerpt)

```
 .....
 for $a in doc("GetQuotesSoapOut.xml")/GetQuotesSoapOut
 <Ontology:StockQuote rdf:ID="StockQuote1">
        <StockQuote:price >
                { fn:string($a/LastUpdated ) }
        </StockQuote:price >
        <StockQuote:volume >
                { fn:string($a/Volume) }
        </StockQuote:volume >
</Ontology:StockQuote>
......

.....
for $b in doc("GetCompanyProfileSoapOut.xml")/GetCompanyProfileSoapOut
 <Ontology:Stock rdf:ID="Stock1">
        <Stock:earnings_per_share>
                { fn:string($b/Est_EPS_F1) }
        </Stock:earnings_per_share>
 </Ontology:Stock>
 .....
```

Lowering mapping: from the Ontology to the helperDecisionRequest message (Excerpt)

```
...
 <ING:helperDecisionRequest>
  <ING:current_price>
     { fn:string($a/StockQuote:price) }
  </ING:current_price>
  <ING:volume>
     {  fn:string($a/StockQuote:volume) }
  </ING:volume>
  <ING:eps>
     { fn:string($a/Stock:earnings_per_share) }
  </ING:eps>
  .....
 </ING:helperDecisionRequest>
```

Figure 12 Binding of partner services - Lifting and Lowering schema mappings

9. Related Work and Discussion

In this section, we present recent data mediation efforts in Web services and discuss past work in the database domain that contribute to interoperability in Web services. The approach presented in this paper for handling message level heterogeneities between interoperating services is based on creating mappings from the message elements to conceptual models (ontologies) and using these mappings for transforming messages at the instance level. A pre-requisite for creating such mappings is matching the WSDL (XML) and ontology schema to identify semantically similar entities between the two schemas; which presents syntactic and model/representational heterogeneities (see Section 3). Past approaches in database integration like (R.J. Miller, 2001), (J. Madhavan, 2001) and (H. Do, 2002) among others, work with heterogeneous models by transforming them into a common representation language and manipulating models in that representation. Our past work on Web service annotation (Patil, 2004) accounts for the difference in expressiveness of XML and ontology schemas by converting both models to a common graph representation to facilitate better matching. However, over a period of time, common representation models or languages have changed. Additionally, transforming to a common model can be lossy (going from a more expressive OWL model to less expressive XML or vice versa), context-sensitive and time consuming. Efforts like (Melnik, 2004) have focused on developing a generic infrastructure that abstracts mappings between models as high level operations which are independent of the data model and application of interest.

In this paper, we have not focused on the automatic or semi-automatic process of matching or the generation of mappings for legitimate reasons. There has been a plethora of work in schema matching and mapping transformations (Rahm, 2001), (Madhavan, 2002). Although the generation of mappings between semantically equivalent, but structurally heterogeneous elements is not a trivial task, it is possible for developers to utilize existing semi-automatic tools and/or manual techniques to generate these mappings. If done manually, heterogeneities and examples defined in Table 1 will hopefully suffice to guide the mapping generation process.

While handling data heterogeneities has been a well researched issue in the context of databases, it has not been investigated very thoroughly in the Web services framework. The WSMO project which coined the term data mediation in the Web services context is most relevant to our work. However, much of their focus so far has been on mediation between ontologies (Mocan, 2005) and not on creating mappings for actual WSDL based services. Some of the recent tools (Oracle, BEA (BEA), Stylus Studio (StylusStudio)) have also focused on creating XQuery based mappings between individual Web services. We believe that our approach which specifies mappings using the available semantics in ontologies will extend the functionality of such tools.

10. Conclusion

In this work, we presented a comprehensive solution for resolving message level heterogeneities between interoperating Web services using pre-defined mappings and extensible elements of existing Web service standards and tools. Although limited in terms of the initial one-time effort required from developers to create and associate mappings, it is important to note that this approach offers great flexibility in terms of extending the available semantics to specify mappings, allowing the re-use of existing tools (Axis 2) and building upon the WSDL standard that the user community is already familiar with. Our data mediation architecture shows how this approach can be integrated into existing Web service based solutions with minimal effort. We recognize that data mediation in Web services is a very challenging problem. This work, albeit not a complete solution to all data mediation issues, is definitely an important step towards realizing interoperability between services. As shown in our evaluation, data mediation is required in most cases for interoperability between services and that our approach facilitates easy runtime binding of services in a Web Process. Our plan for future work includes characterizing what mapping languages are well suited for representing lifting and lowering schema mappings in Web services. Formalizing these mappings and extending the system to support inter-ontology mappings is an additional focus.

APPENDIX A

Output messages of two plug-n-play Stock Quote services

1. Output message of http://ws.strikeiron.com/BasicRealTimeQuotes?wsdl

```
<message name="GetOneQuoteSoapOut">
  <part name="Symbol" type="s:string" wssem:modelReference="Ontology0#Stock.stockSymbol"/>
  <part name="CUSIP" type="s:string" wssem:modelReference="Ontology0#Stock.CUSIP"/>
  <part name="CIK" type="s:string" wssem:modelReference="Ontology0#Stock.CIK"/>
  <part name="Name" type="s:string" wssem:modelReference="Ontology0#Stock.shareOf"/>
  <part name="Date" type="s:string" />
  <part name="Time" type="s:string" />
  <part name="Last" type="s:double" wssem:modelReference="Ontology0#RealTimeStockQuote.price"/>
  <part name="Quantity" type="s:int" wssem:modelReference="Ontology0#RealTimeStockQuote.quantity"/>
  <part name="ChangeFromPrevious" type="s:double" />
  <part name="PercentChangeFromPrevious" type="s:double" wssem:modelReference="Ontology0#ChangeMeasure.percent"/>
  <part name="Open" type="s:double" wssem:modelReference="Ontology0#Stock.open"/>
  <part name="ChangeFromOpen" type="s:double" wssem:modelReference="Ontology0#Stock.changeInValue"/>
  <part name="PercentChangeFromOpen" type="s:double" />
  .....
  <part name="Highest" type="s:double" wssem:modelReference="Ontology0#DailyChange.high"/>
  <part name="Lowest" type="s:double" wssem:modelReference="Ontology0#DailyChange.low"/>
  <part name="Rank" type="s:string" />
 </message>
```

2. Output message of ws.cdyne.com/delayedstockquote/delayedstockquote.asmx?wsdl

```
<wsdl:message name="GetQuoteSoapOut">
   <wsdl:part name="StockSymbol" type="s:string" wssem:modelReference="Ontology0#Stock.stockSymbol"/>
   <wsdl:part name="LastTradeAmount" type="s:decimal" wssem:modelReference="Ontology0#RealTimeStockQuote.price"/>
   <wsdl:part name="LastTradeDateTime" type="s:dateTime" />
   <wsdl:part name="StockChange" type="s:decimal" wssem:modelReference="Ontology0#Stock.changeInValue"/>
   <wsdl:part name="OpenAmount" type="s:decimal" wssem:modelReference="Ontology0#Stock.open"/>
   <wsdl:part name="DayHigh" type="s:decimal" wssem:modelReference="Ontology0#DailyChange.high"/>
   <wsdl:part name="DayLow" type="s:decimal" wssem:modelReference="Ontology0#DailyChange.low"/>
   <wsdl:part name="StockVolume" type="s:int" />
   <wsdl:part name="MktCap" type="s:string" />
   <wsdl:part name="PrevCls" type="s:decimal" />
   <wsdl:part name="ChangePercent" type="s:string" wssem:modelReference="Ontology0#DailyChange.percent"/>
   <wsdl:part name="FiftyTwoWeekRange" type="s:string" wssem:modelReference="Ontology0#_52WeekChange"/>
   <wsdl:part name="EarnPerShare" type="s:decimal" />
   <wsdl:part name="PE" type="s:decimal" />
   <wsdl:part name="CompanyName" type="s:string" wssem:modelReference="Ontology0#Stock.shareOf"/>
   <wsdl:part name="QuoteError" type="s:boolean" />
 </wsdl:message>
```

REFERENCES

UMBC's Cobra Ontology Viewer http://cobra.umbc.edu/eclipse/
jUDDI http://ws.apache.org/juddi/
METEOR-S: Semantic Web Services and Processes.
XML Schema Mapping - Stylus Studio.
OMG, Unified Modeling language (UML) http://www.omg.org/technology/documents/formal/uml.htm
ebXML Core Component Dictionary.
WSMO, Web Services Modeling Ontology.
BEA AquaLogic http://www.bea.com/content/news_events/white_papers/BEA_AQL_Family_ds.pdf
BPEL. Business Process Execution Language for Web Services version 1.1, .
Crub´ezy, M. and Musen, M.A., 2003, *Ontologies in support of problem solving*. Springer
D. Calvanese, G. Giacomo and Lenzerini., M. Ontology of integration and integration of ontologies *Description Logic Workshop* 2001, 10-19.
Dou, D., McDermott, D. and Qi, P. Ontology translation on the semantic web *ODBASE*, 2003.
ebXML http://www.ebxml.org/
Evaluations Evaluation - interoperability of Web services http://lsdis.cs.uga.edu/~meena/ICWS06/Eval.html
GeocodeService StrikeIron US Geocode Information http://ws.strikeiron.com/USGeocoding?WSDL

Gomadam, K. and al., e., 2005, Radiant: A tool for semantic annotation of Web Services. in *The Proceedings of the 4th International Semantic Web Conference (ISWC 2005)* (2005).

H. Do and E. Rahm, 2002, COMA - A System for Flexible Combination of Schema Matching Approaches, *VLDB*. 610-621.

J. Madhavan, P. Bernstein and Rahm, E., 2001, Generic Schema Matching with Cupid. in *27th Int. Conf. on Very Large Data Bases*, (2001).

Kalfoglou, Y. and Schorlemmer, M., 2003, Ontology mapping: the state of the art: The Knowledge Engineering Review, *18(1)*. 1--31.

Kashyap, V. and Sheth, A., 1996, Semantic and schematic similarities between database objects: a context-based approach, *VLDB Journal*.

Kementsietsidis A. , Arenas M. and J., M.R. Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues *SIGMOD* 2003.

Kim, W., Choi, I., Gala, S.K. and Scheevel., M., 1993, On Resolving Schematic Heterogeneity in Multidatabase Systems., *Distributed and Parallel Databases*.

Litwin, W. and Abdellatif, A., 1986, Multi-database Interoperability, *IEEE Computer*, *19(12)*. 10-18.

Madhavan, J., Bernstein, P., Domingos, P. and Halevy, A. Representing and Reasoning about Mappings between Domain Models, *The Eighteenth National Conference on Artificial Intelligence*, Edmonton, Canada, 2002.

Maedche A , Motik B. , Silva N. and R., V. MAFRA - A MApping FRAmework for Distributed Ontologies *13th international Conference on Knowledge Engineering and Knowledge Management*, 2002.

Mandell, D.J. and McIlraith, S.A. Adapting {BPEL4WS} for the Semantic Web: The Bottom-Up Approach to Web Service Interoperation *Proceedings of the Second International Semantic Web Conference*, 2003.

Medjahed, B., Bouguettaya, A. and Elmagarmid, A.K., 2003, Composing Web services on the Semantic Web, *VLDB J. 12(4): 333-351*.

Melnik, S. Generic Model Management: Concepts and Algorithms, Ph.D. Dissertation, University of Leipzig, Springer LNCS 2967, 2004.

Mena, E., Kashyap, V., Sheth, A. and Illarramendi, A., 1996, OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. in *CoopIS*, (1996).

Mocan, A. and Cimpian, E. D13.3v0.2. WSMX Data Mediation http://www.wsmo.org/TR/d13/d13.3/v0.2/20051011/d13.3v0.2_20051011.pdf

Onose, N. and Simeon, J., 2004, XQuery at your web service. in *WWW*, (2004).

P. Bouquet, F. Giunchiglia, F. van Harmelen, L. Serafini and H. Stuckenschmidt, 2003, C-OWL: Contextualizing Ontologies. in *ISWC*, (2003), 164--179.

Patil, A., Oundhakar, S., Sheth, A. and Verma, K., 2004, METEOR-S Web service Annotation Framework. in *WWW*, (2004), 553-562.

PhoneLookupService. StrikeIron Reverse Phone Lookup.

R.J. Miller, M.A. Hernandez, L.M. Haas, L. Yan, C. T. Howard Ho, Fagin, R. and Popa, L., 2001, The Clio project: managing heterogeneity, *SIGMOD 30(1)*. 78--83.

Radiant SAWSDL / WSDL-S Annotation Tool http://lsdis.cs.uga.edu/projects/meteor-s/downloads/index.php?page=1

Rahm, E. and Bernstein, P., 2001, A survey of approaches to automatic schema matching, *VLDB Journal*.

Rajasekaran, P., Miller, J.A., Verma, K. and Sheth, A.P., 2004, Enhancing Web Services Description and Discovery to Facilitate Composition, *Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition*.

RDF Resource Description Framework http://www.w3.org/RDF/

RosettaNet eBusiness Standards for the Global Supply Chain http://www.rosettanet.org/

S.B. Davidson, A. Kosky and Buneman, P., 1995, Semantics of Database Transformations: Semantics in Databases. 55-91.

SAWSDL Semantic Annotations for Web Services Description Language Working Group http://www.w3.org/2002/ws/sawsdl/

SAXON SAXON - The XSLT and XQuery Processor http://saxon.sourceforge.net/

Sheth, A. and Kashyap, V., 1992, So far (schematically) yet so near (semantically). in *Conference on Semantics of Interoperable Database Systems.*, (1992).

Sheth, A., 1998, Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics, *Interoperating Geographic Information Systems*. 5-30.

Sheth, A.P. and Larson, J.A., 1990, Federated database systems for managing distributed, heterogeneous, and autonomous databases, *ACM Computing Surveys*.

Sivashanmugam, K., Verma, K., Sheth, A. and Miller, J., 2003, Adding Semantics to Web Services Standards. in *ICWS*, (2003).

Sivashanmugam, K., Miller, J., Sheth, A. and Verma, K., 2004a, Framework for Semantic Web Process Composition, *International Journal of Electronic Commerce*.

Sivashanmugam, K., Miller, J., Sheth, A. and Verma, K., 2004b, Framework for Semantic Web Process Composition, *IJEC*, *Vol. 9(2)* pp. 71-106.

StylusStudio XML Editor, XML Data Integration, XML Tools, Web Services and XQuery http://www.stylusstudio.com/

SUMO Suggested Upper Merged Ontology http://ontology.teknowledge.com/

UDDI. Universal Description, Discovery and Integration.

UDDIregistries, Public UDDI registries.

Verma, K. Configuration and Adaptation of Semantic Web Processes *Department of Computer Science*, The University of Georgia, 2006.

WSDL-S Web Service Semantics http://www.w3.org/Submission/WSDL-S/

Zeng, L., Benatallah, B., Dumas, M., Kalagnanam, J. and Sheng, Q.Z., 2003, Quality driven web services composition. in *WWW: 411-421*, (2003).

ABOUT THE AUTHORS

Meenakshi Nagarajan (http://lsdis.cs.uga.edu/~meena) is pursuing a PhD in Computer Science at the University of Georgia under the direction of Prof. Amit Sheth. Her interests lie in the space of Matching, Mapping and Disambiguation of structured, semi-structured and unstructured data. She is a co-author of WSDL-S, which is the key driver for W3C's candidate recommendation SAWSDL. She received her undergraduate degree from BITS, Pilani India and will be a part of the Kno.e.sis Center (http://knoesis.org) at the Wright State University starting 2007.

Dr. Kunal Verma is a researcher with Accenture Technology Labs. He has published a number of research papers in the areas of SOA, Semantic Web services and processes, adaptive Web processes and XML Databases. He is a co-author of WSDL-S, which is the primary input for W3C's upcoming candidate recommendation SAWSDL. He graduated with a Ph.D in Computer Science from the University of Georgia.

Prof. Amit Sheth is an educator, researcher and entrepreneur. He is the LexisNexis Eminent Scholar and the director of Kno.e.sis Center (http://knoesis.org) at the Wright State University. He is a fellow of the IEEE. He is well published and cited in the areas of information integration, workflow management and Semantic Web. Earlier, he founded and directed the LSDIS lab at University of Georgia, and worked in R&D positions at Bellcore (Telcordia), Unisys and Honeywell. He received his graduate education at Ohio State University and undergraduate education at BITS, Pilani, India.

John A. Miller is a Professor of Computer Science at the University of Georgia and has also been the Graduate Coordinator for the department for 9 years. His research interests include database systems, simulation, bioinformatics and Web services. Dr. Miller received the B.S. degree in Applied Mathematics from Northwestern University in 1980 and the M.S. and Ph.D. in Information and Computer Science from the Georgia Institute of Technology in 1982 and 1986, respectively. He is the author of over 100 technical papers in the areas of database, simulation, bioinformatics and Web services. He is an Associate Editor for ACM Transactions on Modeling and Computer Simulation and IEEE Transactions on Systems, Man and Cybernetics as well as a Guest Editor for the International Journal in Computer Simulation and IEEE Potentials.