Semantic Interoperability of Web Services – Challenges and Experiences

Meenakshi Nagarajan, Kunal Verma, Amit P. Sheth, John Miller, Jon Lathem
LSDIS Lab, Department Of Computer Science, University of Georgia, Athens GA, USA
{bala,verma,sheth,jam,lathem}@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. We present a data mediation architecture using the extensibility features of WSDL and the popular SOAP engine, Axis 2.

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 [43], composition [37], [20], [47] and execution [5] 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 [3], interoperability among databases with heterogeneous schemas has been a well researched issue [18] [35]. 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 [23] project, which aims to define semantics for the complete lifecycle of semantic Web processes.

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 [15]. Our approach uses the support for data mapping provided in WSDL-S [44], which is a W3C acknowledged member submission for semantic Web services. The aim is to provide a solution to the problem of Web service interoperation by making incremental changes to Web services tools. Since WSDL-S builds upon existing Web services standards (WSDL), it also allows us to use the extensibility support provided by Axis 2 to implement data mediation. This paper has the following contributions:

- We present a comprehensive, practical approach for resolving data heterogeneities between Web services.
- 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).

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 [39]) that returns an address for a listed telephone number. The second Web service (available at [40]) 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.

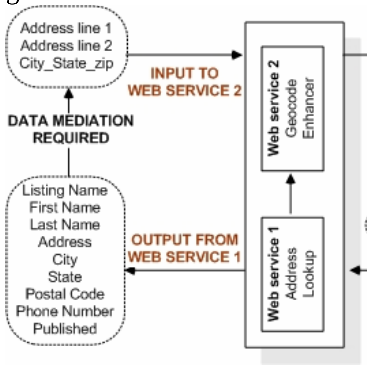


Figure 1 Process showing need for Data Mediation

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.

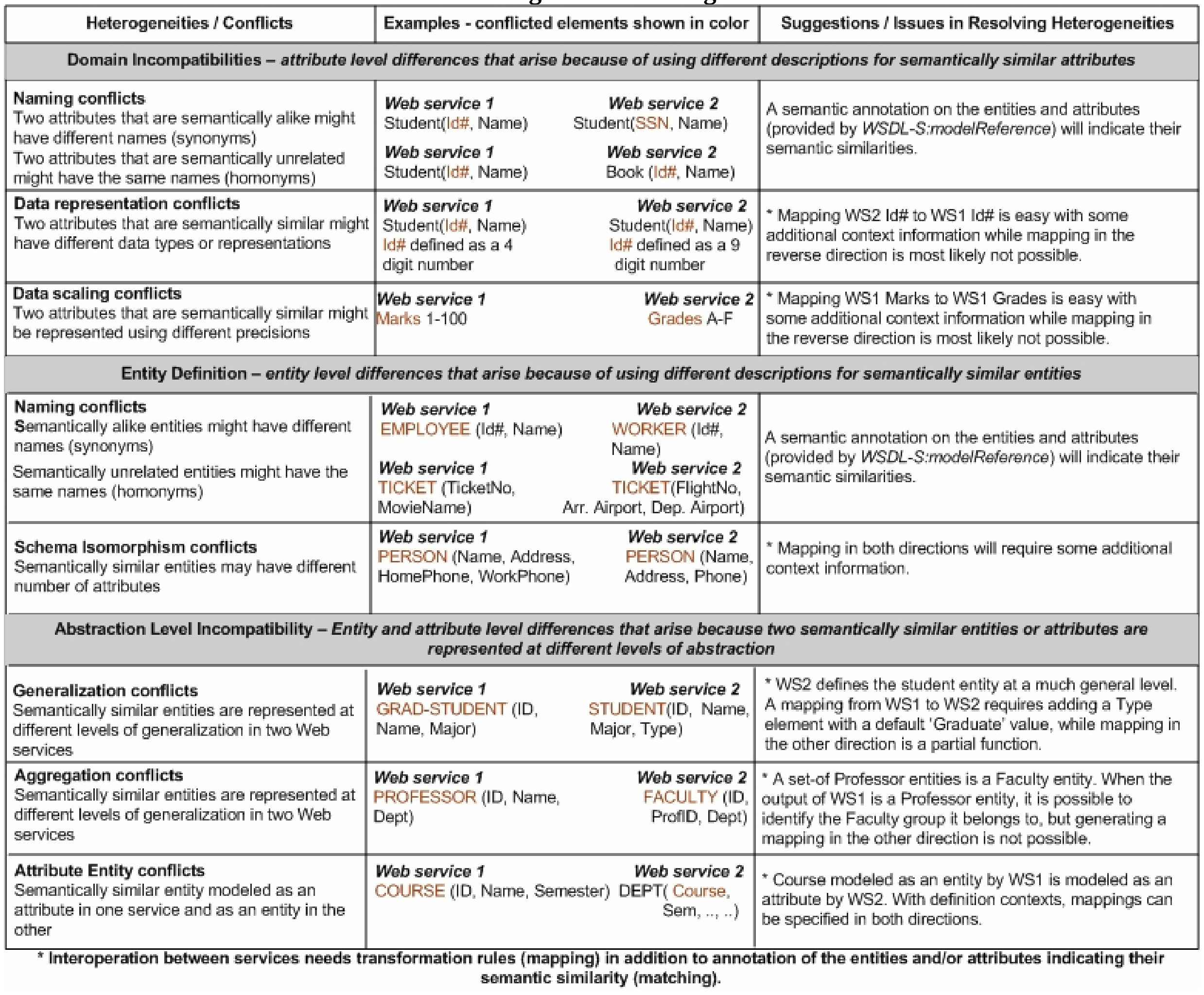
3. Message-Level Heterogeneities

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 8.

In both databases and Web services, automating the process of matching and mapping is hard due to heterogeneities at the following levels [36], [35]: *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 [17], [15] include 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) **Domain 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. Table 1 illustrates each of these conflicts by using interoperating Web services and suggests how one might resolve these conflicts using semantic annotations and/or mappings between the services.

Table 1 Message Level Heterogeneities



Heterogeneities / Conflicts	Examples - conflicted elements shown in color	Suggestions / Issues in Resolving Heterogeneities
Domain Incompatibilities – ***attribute level 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 *WSDL-S: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 Definition – ***entity level 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 *WSDL-S: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 Semantically similar entities are represented at different levels of generalization in two Web services	***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).**

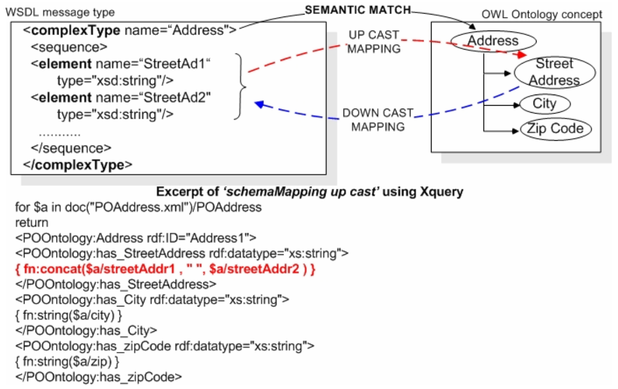
In addition to matching and mapping, the representation of mappings is also of significant concern. The expressiveness of the mapping language can dictate to a large extent, the types of heterogeneities that can be resolved. Some of the past approaches to representing mappings have been *queries or views* [7], XQuery, XSLT; *mapping tables* [2]; *bridging axioms* [19], [8]; as *instances in an ontology of mappings* [1], [6]; *languages* [34], [28], [17], etc. In this work, we use a popular representation for mappings in Web services, *XQuery and XSLT* [46], [27] and use WSDL-S 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 the event that these do not suffice, the developer has the flexibility of using any mapping language since WSDL-S is agnostic to the mapping representation used. Since our implementation for data mediation exploits this feature of WSDL-S, it is also independent of the mapping or conceptual model representation used.

4. WSDL-S

WSDL-S [44], a W3C member submission for Semantic Web services provides a mechanism to annotate the capabilities and requirements of Web services (described using WSDL) with semantic concepts defined in an external domain model. 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 [32] and UML [26]. Of the six extensibility elements and attributes provided in WSDL-S, the *modelReference* and *schemaMapping* extensibility attributes are most relevant to this work. The *modelReference* extension attribute is used to specify the association between a WSDL element and a concept in some semantic model. It can be added to a complex type, an element, an operation and their preconditions and effects. The *schemaMapping* extension attribute is added to WSDL XSD elements and complex types, for handling structural differences between the schema elements of a Web service and their corresponding semantic model concepts.

5. Proposed Data Mediation Approach

Support for data mediation in WSDL-S is provided by having the developer associate mappings (created either manually or using semi-automatic tools) using the *'schemaMapping'* attribute 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 2. In addition to a mapping from the Web service message element to the ontology concept, also called the *'up cast'*, an additional mapping from the ontology concept to the message element, called the *'downcast'*, 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 WSDL-S is agnostic to the domain model representation language.



```
<complexType name="Address">
  <sequence>
  <element name="StreetAd1"
      type="xsd:string"/>
  <element name="StreetAd2"
      type="xsd:string"/>
   ...........
  </sequence>
</complexType>
```

```
for $a in doc("POAddress.xml")/POAddress
return
<POOntology:Address rdf:ID="Address1">
<POOntology:has_StreetAddress rdf:datatype="xs:string">
{ fn:concat($a/streetAddr1 , " ", $a/streetAddr2 ) }
</POOntology:has_StreetAddress>
<POOntology:has_City rdf:datatype="xs:string">
{ fn:string($a/city) }
</POOntology:has_City>
<POOntology:has_zipCode rdf:datatype="xs:string">
{ fn:string($a/zip) }
</POOntology:has_zipCode>
```

Figure 2 *'Up cast'* and *'Downcast' schemaMappings*

Data transformation proceeds in three steps as shown in Figure 3. In the first step (1), the output message of Web Service 1 (WS1) is transformed to the OWL concept to which it is mapped *(up cast)*; next, the OWL concept is transformed to the input message of Web service 2 (WS2) (3) *(downcast)*. 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 8 for a discussion on ontology matching and mapping). As we can see, 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.

The current draft of the specification provides only one '*schemaMapping*' attribute for associating mappings from the Web service element to the ontology concept *(up cast)*. Since there are cases when the automatic generation of the reverse *'downcast'* mappings (given only the source and target schemas and the *'up cast'* mapping) might not be possible, we have proposed the addition of the *'schemaMapping downcast'* extension attribute to WSDL-S. For example, as in Figure 2, the concatenation of 'streetAddress1' and 'streetAddress2' (parts of a WSDL message element) to 'streetAddress' (part of an ontology concept) is quite straightforward and is shown as an XQuery in the figure, while the generation of the reverse mapping, i.e., splitting 'streetAddress' into two entities is hard to automate (i.e., It is hard to automate where one would split a street Address to form two streetAddress1 and streetAddress2)

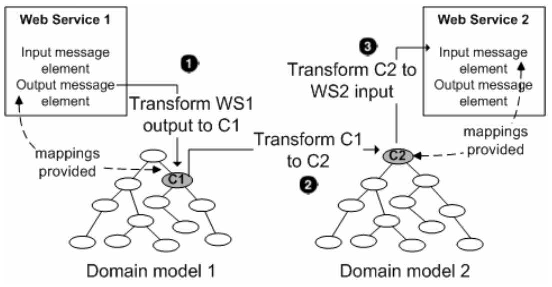


Figure **3 Data transformation using WSDL-S**

5.1. System Architecture

The general philosophy of the METEOR-S project, as outlined in [38], 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. 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 4 consists of a main METEOR-S middleware component implemented as modules on an Axis2 server. The actual METEOR-S middleware component [23] 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 WSDL-S by annotating the WSDL file with semantic concepts from an ontology (using a tool like [10]). The *up cast* and *downcast* mappings from the Web service message elements to the semantic concepts should be created and associated using the *'schemaMapping'* functions.
- 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 [23]), performs the *up cast* and *downcast* 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 [16] is an open source XQuery/XSLT processor that we use to process the mappings represented using XQuery/XSLT.

5.2. 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; although users can emulate a process by chaining invocation of services in Java. Both Web services are described using WSDL-S 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.

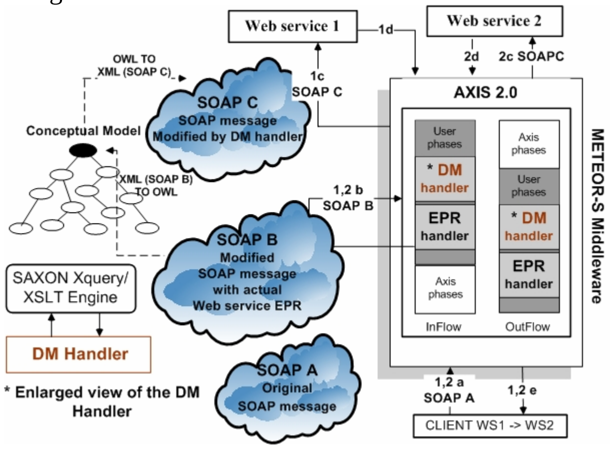


Figure 4 Data Mediation System Architecture

In Figure 4, 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.

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 Figure 3):

i. Using the namespaces in the SOAP message and the logical to physical map in the *EPR* module, the WSDL-

S file is accessed to get the '*schemaMapping up cast*' 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 WSDL-S of the actual Web service is accessed to get the *'schemaMapping downcast'* mapping provided on the input message element of the Web service being invoked (Web service 2).
iii. The *'schemaMapping up cast'* 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 *schemaMapping 'downcast'* 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.

6. 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.

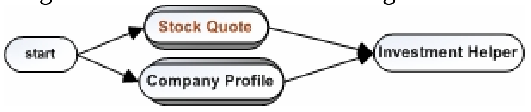


Figure 5 Investment Assistance Process

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 [12]. 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.

As we can see, 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 [42] ontology and available at [25]) 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 [25].

7. 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 [30], [13] and [11] 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 [29] 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 [21] 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 [31], [19]. 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.

Additionally, 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 [33] and ebXML CCD [9]) 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, [22], [7] and [14] discuss this problem in different contexts and provide useful insights. While we plan to extend our approach to handle multiple ontology matching and mapping, our solution is still is useful as number of popular specifications/ontologies (ebXML, RosettaNet, OAGIS, etc.) are currently being used for interoperability between business partners.

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 [45] 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 [24] and not on creating mappings for actual WSDL based services. Some of the recent tools (Oracle, BEA [4], Stylus Studio [41]) 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.

8. Conclusion

In this work, we present 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. Our plan for future work includes incorporating a framework for inter-ontology mappings and creating tools with support for semi-automated matching and mapping.

9. References

[1] A Maedche, B. Motik, N. Silva and R. Volz, MAFRA.

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

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

[4] BEA AquaLogic http://www.bea.com/content/news_events/white_papers/BEA_AQL_Family_ds.pdf

[5] http://www-128.ibm.com/developerworks/library/specification/ws-bpel/ Business Process Execution Language for Web Services

[6] M. Crub´ezy and M. A. Musen *Ontologies in support of problem solving*. Springer, 2003.

[7] D. Calvanese, G. Giacomo and M. Lenzerini. Ontology of integration and integration of ontologies *Description Logic Workshop* 2001, 10-19.

[8] D. Dou, D. McDermott and P. Qi. Ontology translation on the semantic web *ODBASE*, 2003.

[9] http://www.ebxml.org/specs/ccDICT.pdf ebXML Core Component Dictionary

[10] K. Gomadam, K. Verma, D. Brewer, A. P.Sheth and J. A. Miller, Radiant: A tool for semantic annotation of Web Services. *ISWC*, 2005.

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

[12] Public UDDI registries http://uddi.microsoft.com;http://uddi.sap.com;www.bindingpoint.com;www.salcentral.com;http://www.strikeiron.com/;www.xmethods.net

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

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

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

[16] SAXON - The XSLT and XQuery Processor http://saxon.sourceforge.net/ M. Kay

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

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

[19] J. Madhavan, P. Bernstein, P. Domingos and A. Halevy. Representing and Reasoning about Mappings between Domain Models, *AAAI*, Edmonton, Canada, 2002.

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

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

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

[23] http://lsdis.cs.uga.edu/projects/meteor-s/ METEOR-S: Semantic Web Services and Processes

[24] D13.3v0.2. WSMX Data Mediation http://www.wsmo.org/TR/d13/d13.3/v0.2/20051011/d13.3v0.2_20051011.pdf

[25] Evaluation - interoperability of Web services http://lsdis.cs.uga.edu/~meena/ICWS06/Eval.html

[26] OMG, Unified Modeling language (UML) http://www.omg.org/technology/documents/formal/uml.htm

[27] N. Onose and J. Simeon, XQuery at your web service. *WWW*, 2004.

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

[29] A. Patil, S. Oundhakar, A. Sheth and K. Verma, METEOR-S Web service Annotation Framework. *WWW*, 2004, 553-562.

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

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

[32] Resource Description Framework. www.w3.org/RDF

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

[34] S.B. Davidson, A. Kosky and P. Buneman, Semantics of Database Transformations: Semantics in Databases. 1995.

[35] A. Sheth, Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. 1998 *Interop. GIS*. 5-30.

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

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

[38] K. Sivashanmugam, K. Verma, A. Sheth and J. Miller, Adding Semantics to Web Services Standards. *ICWS*, 2003.

[39] http://ws.strikeiron.com/ReversePhoneLookup?WSDL StrikeIron Reverse Phone Lookup

[40] http://ws.strikeiron.com/USGeocoding?WSDL StrikeIron US Geocode Information

[41] http://www.stylusstudio.com/ Stylus Studio – XML Editor, XML Data Integration, XML Tools, Web Services and XQuery

[42] http://ontology.teknowledge.com/ SUMO - Suggested Upper Merged Ontology

[43] http://www.uddi.org/about.html Universal Description, Discovery and Integration

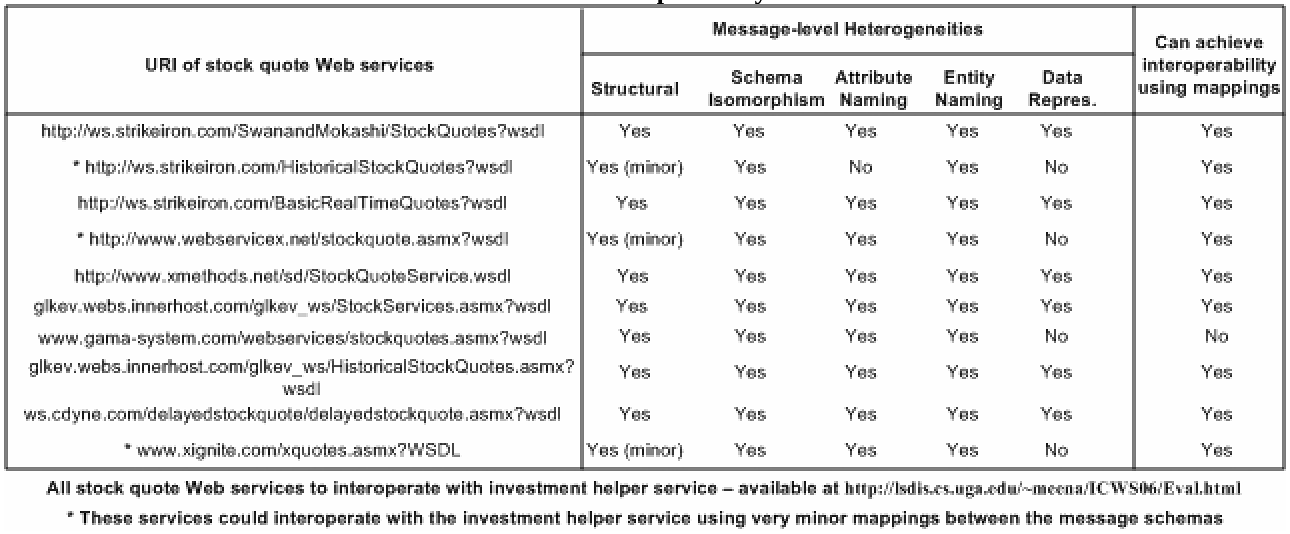
[44] http://www.w3.org/Submission/WSDL-S/ WSDL-S, Web Service Semantics

[45] WSMO, Web Services Modeling Ontology. http://www.wsmo.org/

[46] http://www.stylusstudio.com/whitepapers/case_for_xquery.pdf XML Schema Mapping - Stylus Studio

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

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**