Chapter 2

SEMANTIC ANNOTATIONS IN WEB SERVICES

Meenakshi Nagarajan
Large Scale Distributed Information Systems (LSDIS) Lab, Department of Computer Science, University of Georgia, GA, USA. – nbmeena@uga.edu

1. INTRODUCTION

"The Semantic Web is a vision: the idea of having data on the Web defined and linked in such a way that it can be used by machines not just for display purposes, but for automation, integration and reuse of data across various applications." (Semantic Web Activity Statement)

Meaningful use of any data requires knowledge about its organization and content. Contextual information that establishes relationships between the data and the real world aspects it applies to is called metadata. In other words, metadata is data that describes information about a piece of data, thereby creating a context in terms of the content and functionality of that data. Domain conceptualizations, ontologies or world models provide agreed upon and unambiguous models for capturing data and metadata to which applications, data providers and consumers can refer. Broadly speaking, there are two kinds of metadata – structural and syntactic metadata. Structural metadata provides information about the organization and structure of some data, e.g. format of the document. Semantic metadata on the other hand, provides information 'about' the data for example the meaning or what the data is about and the available semantic relationships from a domain model in which the data is defined.

The key aspect behind the realization of the Semantic Web vision is the provision of metadata and the association of metadata with web resources. The process of associating metadata with resources (audio, video, structured text, unstructured text, web pages, images etc) is called annotation and semantic annotation is the process of annotating resources with semantic metadata.

Semantic annotations can be coarsely classified as being formal or informal. Formal semantic annotations, unlike informal semantic annotations follow representation mechanisms, drawing on conceptual models represented using well-defined knowledge representation languages. Such machine processable formal annotations on web resources can result in vastly improved and automated search capabilities, unambiguous resource discoveries, information analytics etc. The annotation of web based resources like text files or digital content is very different from the annotation of Web services. In this chapter, we will explore the nature of semantics associated with the Web services and different aspects of semantic annotation of Web resources and Web services in particular.

1.1 Generic Semantic Annotation Architecture

Semantic annotation of resources supported by an existing world model (the ontology schema that provides an agreed upon and unambiguous model for capturing data and metadata) and knowledge base (ontology instances) follows three primary steps: entity identification, entity disambiguation and annotation. These three steps vary depending on the kind of resource one is trying to annotate.

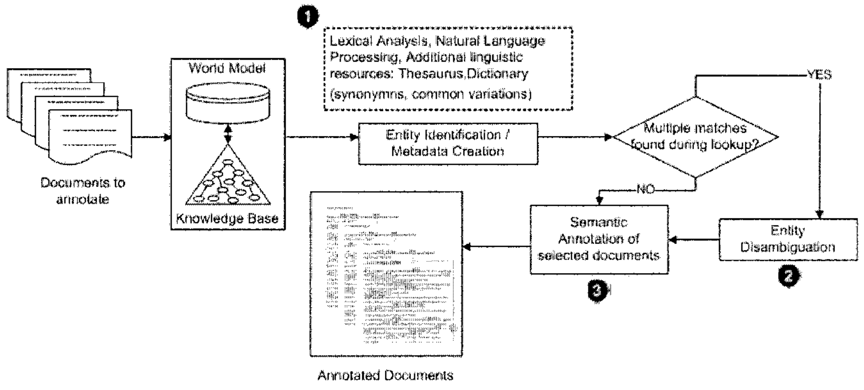

1
Lexical Analysis, Natural Language Processing, Additional linguistic resources: Thesaurus,Dictionary (synonymns, common variations)
World Model
Knowledge Base
Documents to annotate
Entity Identification / Metadata Creation
Multiple matches found during lookup?
YES
NO
Semantic Annotation of selected documents
3
Entity Disambiguation
2
Annotated Documents

Figure 2-1. Semantic Annotation of documents

For example, the process of identifying entities that need to be annotated from a textual document is different from the process of identifying potential entities from experimental data. The underlying idea however remains the same. In this section, we will briefly cover the three steps involved in the semantic annotation of a resource. For the sake of simplicity, the resource considered for annotation is a text document and the semantics are brought in using a single ontology; although there is nothing that prevents the user from using multiple ontologies.

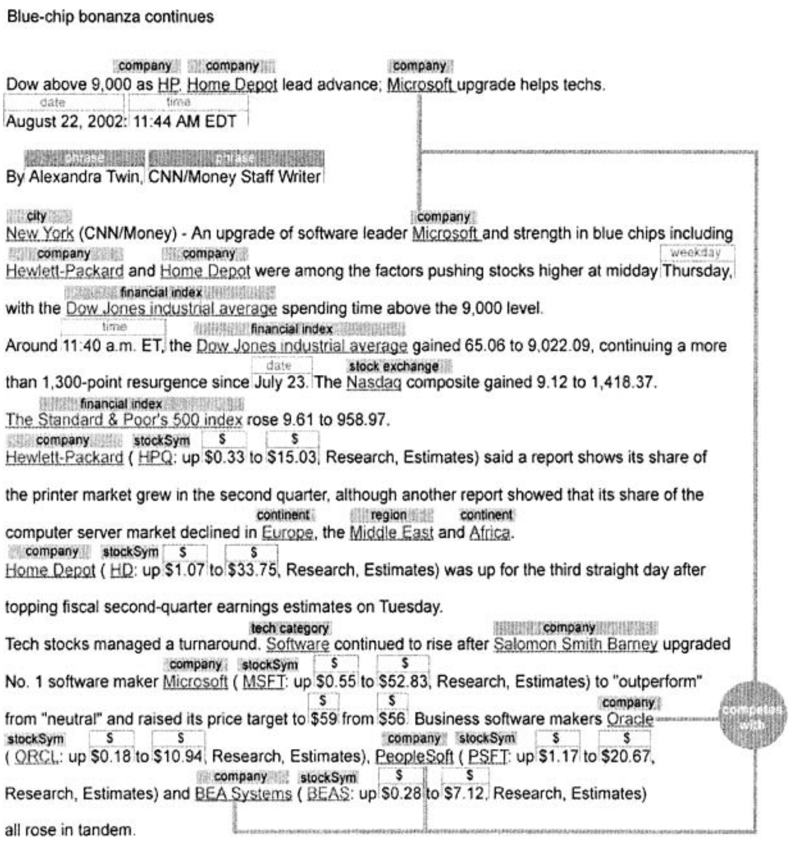
Figure 2-1 shows the process of semantically annotating a set of documents with the semantics provided by a world model (ontology schema) and a knowledge base (ontology instances).

1.1.1 Entity Identification

The process of entity identification (shown as step 1 in Figure 2-1), involves extracting useful information from a document with the help of rule-based grammars, natural language processing techniques, user-defined templates or wrappers, etc. In addition to the above technologies, ontology-driven extraction of entities also uses the populated ontology (instance level information, also called the knowledge base that is populated using the ontology schema) to extract specific instances of different classes. The approach shown in Figure 2-1 uses a combination of an existing ontology and knowledge base, lexicons and natural language processing techniques.

When an entity is identified in a document, a check is performed to see if the entity exists as an instance in the knowledge base. Variations of the entity like the presence of prefixes or suffixes (such as Jr., Dr., III), common abbreviations (such as US for United States), synonyms, similar strings (accounting for mis-spellings in the document) etc. are also taken into consideration while looking for corresponding instances in the knowledge base. Figure 2-2 shows identified entities in a CNN business article and the corresponding classes from a Stock ontology. Entities of interest are underlined (in blue) and the ontology classes they are associated with are shown in grey. For example, *New York* is an instance of class *City*; *Microsoft* is an instance of class *Company* etc.

In addition to making the process of entity identification more scalable and specialized to a domain, using a knowledge base also allows users to see relationships (already in knowledge base) between identified entities not present in the document itself. For example, the fact that Microsoft and Oracle (see Figure 2-2) are competitors is not in the document and is available to the user only because it was present in the knowledge base.


Blue-chip bonanza continues

Dow above 9,000 as HP, Home Depot lead advance; Microsoft upgrade helps techs.

August 22, 2002: 11:44 AM EDT

By Alexandra Twin, CNN/Money Staff Writer

New York (CNN/Money) - An upgrade of software leader Microsoft and strength in blue chips including Hewlett-Packard and Home Depot were among the factors pushing stocks higher at midday Thursday, with the Dow Jones industrial average spending time above the 9,000 level.

Around 11:40 a.m. ET, the Dow Jones industrial average gained 65.06 to 9,022.09, continuing a more than 1,300-point resurgence since July 23. The Nasdaq composite gained 9.12 to 1,418.37. The Standard & Poor's 500 index rose 9.61 to 958.97.

Hewlett-Packard (HPQ: up $0.33 to $15.03, Research, Estimates) said a report shows its share of the printer market grew in the second quarter, although another report showed that its share of the computer server market declined in Europe, the Middle East and Africa.

Home Depot (HD: up $1.07 to $33.75, Research, Estimates) was up for the third straight day after topping fiscal second-quarter earnings estimates on Tuesday.

Tech stocks managed a turnaround. Software continued to rise after Salomon Smith Barney upgraded No. 1 software maker Microsoft (MSFT: up $0.55 to $52.83, Research, Estimates) to "outperform" from "neutral" and raised its price target to $59 from $56. Business software makers Oracle (ORCL: up $0.18 to $10.94, Research, Estimates), PeopleSoft (PSFT: up $1.17 to $20.67, Research, Estimates) and BEA Systems (BEAS: up $0.28 to $7.12, Research, Estimates) all rose in tandem.

Figure 2-2. Entity identification in an unstructured document (Hammond et al. 2002)

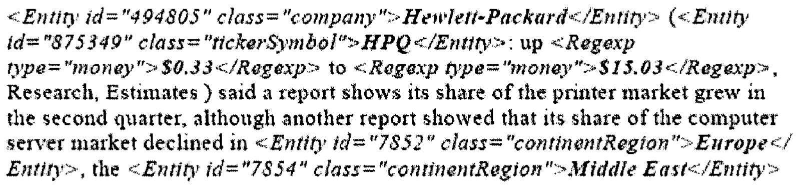
1.1.2 Entity Disambiguation

Very often it is possible that for an entity identified in the document, there are multiple references to it in the knowledge base. For example, for an instance John Smith identified in a document, there could be two instances of John Smith in the knowledge base, one a financial analyst and the other the CEO of a company. The information pertaining to the entity John Smith in the document might not exactly correspond to the information available for the same entity in the knowledge base. For example, the document might not explicitly mention John Smith as the CEO of the company but could be an article about the strategies of the company that John Smith is a CEO of. In such a case, sophisticated methods are required to glean the context in which John Smith is mentioned in the document. Different data sources have different ways of representing the same real world entity. Variations in representation usually arise due to incorrect spellings, use of abbreviations, different naming conventions, naming variations over time, etc. Entity disambiguation (shown as step 2 in Figure 2-1) is the process of identifying when different references correspond to the same real world entity. Entity disambiguation is crucial to basic functionalities like database/ontology integration, population, and to many information management system applications (Blume 2005). A multitude of approaches exist to disambiguate entities depending on the nature of the data source and the level of accuracy required; (Kalashnikov et al. 2005, Dong et al. 2005, Han et al. 2004) represents a small sample of the literature.

In this example setting, the need is to disambiguate the entity identified in the document and the multiple candidate references found in the ontology. Extensive use of context information provides the best evidence for reconciliation decisions. Context of an entity mentioned in a document could be defined in terms of the context of the document, the document's classification in a subject hierarchy etc. to glean what the document is talking about. Context of an entity in a knowledge base could be defined in terms of the values for attributes an entity has and the relationships it participates in. For example, if for the entity "BEAS" appearing in the document, there are two instances in the ontology appearing in the contexts "Bureau of Elder and Adult Services BEAS: an organization" and "BEAS: stock symbol for BEA Systems"; gleaning the context in which "BEAS" appears in the document i.e. associated with BEA Systems can help disambiguate the two references in the ontology. Entity disambiguation is a data and engineering intensive process and usually requires some amount of user involvement.

1.1.3 Annotation

After the entity disambiguation process (in the presence of ambiguities), the next step is to associate semantic metadata to the entities in the document through the process of annotation. Typically intended for use by humans and agents, these annotations are represented using W3C recommended standard representation languages like RDF (Resource Description Framework) / OWL (Web Ontology Language, OWL). Figure 2-3 shows sample metadata for a few entities in the document shown in Figure 2-2. The annotation made in XML (Extensible Markup Language (XML)) shows the entity *'Hewlett-Packard'* is an instance of class *'company'*, *'HPQ'* is a *'tickerSymbol'* etc.



```
<Entity id="494805" class="company">Hewlett-Packard</Entity> (<Entity
id="875349" class="tickerSymbol">HPQ</Entity>: up <Regexp
type="money">$0.33</Regexp> to <Regexp type="money">$15.03</Regexp>,
Research, Estimates ) said a report shows its share of the printer market grew in
the second quarter, although another report showed that its share of the computer
server market declined in <Entity id="7852" class="continentRegion">Europe</
Entity>, the <Entity id="7854" class="continentRegion">Middle East</Entity>
```

Figure 2-3. Sample Semantic Annotation in XML

Metadata Enhancement: In the process of identifying entities in the document, it is possible that we find values for attributes or relationships that were not previously present in the knowledge base. Enhancing the existing metadata could be as simple as entering values for attributes, in which case they could be automated; or as complex as modifying the underlying schema, in which case some user involvement might be required.

1.2 Semantic Annotation Applications

Several efforts have been made towards building scalable, automatic semantic annotation platforms. Most of these systems focus on manual and semi-automatic tooling to improve the productivity of a human annotator rather than on fully automated methods. However, even with machine assistance, annotation of content is a difficult, time consuming and error-prone task.

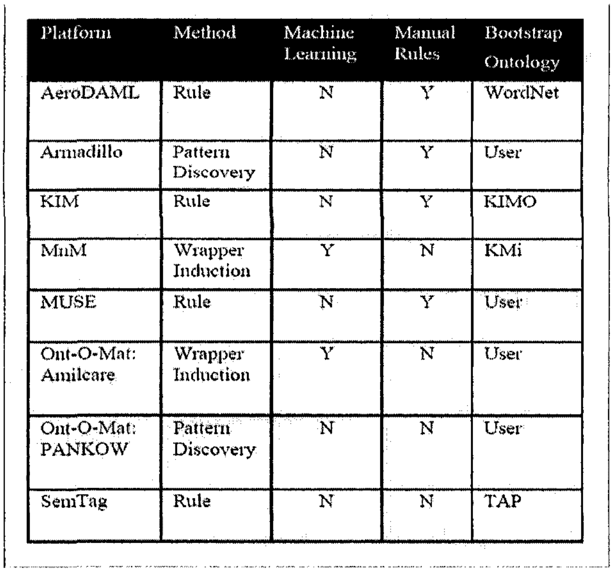
Besides semantic tagging of content, a number of applications also provide storage of annotations and ontologies, user interfaces, access APIs, and features to fully support annotation usage. The most interesting aspect of these applications is the variety of information extraction techniques used. Rules, discovering patterns, machine learning and bootstrapping from taxonomies or ontologies are some techniques used. Examples of such efforts include SemTag (Dill et al. 2003), SHOE (The SHOE Knowledge Annotator), AeroDAML (Kogut et al. 2001), SEE (Hammond et al. 2002), OntoAnnotate (Staab et al. 2001), COHSE (Goble et al. 2001), CREAM (Handschuh et al. 2002), Annotea (Kahan et al. 2002), KIM (Popov et al. 2003) etc. The page on (Annotation Tools) also lists some available tools. Table 2-1 shows a comparison of some tools on the basis of the technology used. In this section, we will briefly describe some applications to give a general idea of the features of annotation frameworks. The reader should refer to Table 2-1 to relate different components of these applications to what has been presented earlier in this chapter.

SemTag is an application written on a platform for large-scale text analytics called Seeker. SemTag performs automated semantic tagging of large corpora using the TAP (Guha et al.) ontology. Also used is a disambiguation algorithm specialized to support ontological disambiguation of large-scale data. Annotations are represented using RDFS (RDF Vocabulary Description Language 1.0: RDF Schema).

SHOE, one the earliest systems for adding semantic annotations to web pages allows users to mark up pages in SHOE (Heflin et al. 1999) guided by ontologies available locally or via a URL. These marked up pages can also be reasoned about by SHOE-aware tools such as SHOE Search (Semantic Search - The SHOE Search Engine).

OntoAnnotate offers comprehensive support for the creation of semantically interlinked metadata by human annotators. In identifying entities in web pages, it uses a combination of the following techniques: wrapper generation, pattern matching and ontology based information extraction based on a shallow text processing engine. Also included in the framework is a document management system that stores annotated documents and their metadata represented in RDF.

Table 2-1. Semantic Annotation Platforms (Reeve et al. 2005)



Platform	Method	Machine Learning	Manual Rules	Bootstrap Ontology
AeroDAML	Rule	N	Y	WordNet
Armadillo	Pattern Discovery	N	Y	User
KIM	Rule	N	Y	KIMO
MnM	Wrapper Induction	Y	N	KMi
MUSE	Rule	N	Y	User
Ont-O-Mat: Amilcare	Wrapper Induction	Y	N	User
Ont-O-Mat: PANKOW	Pattern Discovery	N	N	User
SemTag	Rule	N	N	TAP

The KIM platform provides a novel Knowledge and Information Management (KIM) infrastructure and services for automatic semantic annotation, indexing, and retrieval of unstructured and semi-structured content. It analyzes texts and recognizes references to entities and tries to match the reference with a known entity. The reference in the document gets annotated with the URI of the entity. KIM is equipped with an upper-level ontology PROTON (PROTON Ontology) and a knowledge base KIM KB (KIM Knowledge Base). Other than automatic semantic annotation, KIM also allows one to perform content retrieval, based on semantic restrictions, as well as querying and modifying the underlying ontologies and knowledge bases.

The work in building ontologies and creating semantic annotations for resources is fundamental to the building of the Semantic Web and is gaining a lot of momentum (Berners-Lee et al. 2001). Besides textual and digital content, the most important Web resources are those that provide 'services'. Such services also called Web services are non-static in nature i.e. they allow one to effect some action or change in the world, such as the purchase of a product. The Semantic Web should enable users and agents to discover, use, compose, and monitor Web-based services automatically. The semantic annotation of Web services is however a completely different ball game than the annotation of other web resources. The semantics associated with Web services need to be formulated in a way that makes them useful to the application of Web services. In (Sheth 2003), four types of semantics are presented for the complete life cycle of a Web process. In the next few sections, we will see how the technology built for the Semantic Web is being applied to enhance Web service descriptions to make the aforementioned tasks possible.

2. SEMANTIC ANNOTATION IN WEB SERVICES

There has been a recent proliferation of Web services as the technology for business process execution and application integration. Although Web services are based on widely accepted standards, the lack of a formal description of the meaning of their functionality and the data exchanged has been a significant roadblock in the realization of integration promises. As the number of Web services increase, it is important to have automated tools to discover and compose Web services. The extent of description available in the current WSDL standard leaves room for ambiguous interpretations of the functionality and data of a Web service. Ambiguity in interpretation hinders the automation of tasks like service discovery, composition, invocation etc. One of the ways the community is working to address these issues is by developing a semantic markup language for Web Services. This section of the chapter discusses different aspects of semantic annotation of Web service elements.

2.1 Annotating a Web Service

Semantically annotating a Web service implies explicating the exact semantics of the Web service data and functionality elements that are crucial towards the use of the Web service. This is done by annotating the Web service elements with concepts in domain models or ontologies. Since ontologies represent an agreed upon view of the modeled domain, any ambiguity in the interpretation of functionality or data of a Web service is eliminated. The purpose of annotating Web services is to enable unambiguous and automated service discovery and composition. For example, two Web services meant for completely different functionalities may use the same data types and names for their operations, inputs and outputs, thus making the interpretation of their functionality ambiguous. To understand what parts of a Web service need to be annotated, it is important to understand the interplay of semantics in the life cycle or their usage in a Web service.

While discovering or composing a Web service, a requestor describes his requirements in terms of the functionality i.e. operations of a Web service, and the data used by them i.e. inputs and outputs. Optional specifications include the preconditions and effects of the operation. Preconditions are requirements that must be met before a Web service operation is invoked and effects are the results of invoking an operation. Semantic annotations are therefore associated with the inputs, outputs, preconditions and effects of an operation element of a Web service. More advanced discovery mechanisms however, consider non-functional aspects of Web services and consumer requirements like quality metrics, reliability, security etc.

The benefits of adding semantics is pervasive in the entire life cycle of a Web process (see Figure 2-4). Developers can use semantic annotations to explicate the capabilities of their Web services (1). Once these Web services are published in the UDDI (Universal Description, Discovery and Integration) (2), a requestor can formulate his requirements in a semantic service template (3) (Sivashanmugam et al. 2003) to discover or compose Web services. A semantic service or process template is an abstract service or process description, where the control flow is created manually and the functionality required is described using terms from a domain model or ontology. Reasoning techniques can be used to compare the requirements in the service template with the capabilities of Web services available in the UDDI (4) to discover services (UDDI Technical White Paper 2000). During composition, the functional aspect of the annotations can be used to create useful service compositions.

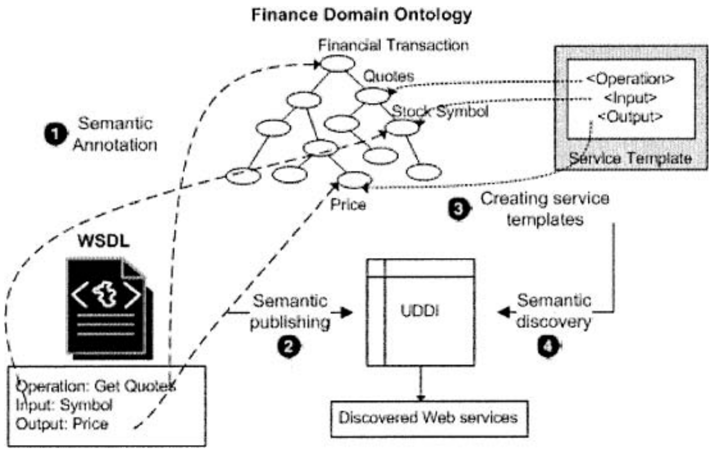


Figure 2-4. Semantics in the life cycle of a Web service

2.2 Four Types of Semantics in Web Services

Table 2-2 illustrates the four types of semantics; data, functional, non-functional and execution semantics associated with Web services and how they relate to the different stages shown in Figure 2-4. Chapter 4 of this book gives an example of how these semantics are modeled in Web services.

Table 2-2. Four types of semantics in Web processes

Type of Semantics	**Description**	**Use**
Data Semantics	Formal definition of data in input and output messages of a Web service	Service discovery and interoperabilit y between Web services
Functional Semantics	Formal definition of the capabilities of a Web service.	Discovery and composition of Web Services
Non-functional Semantics	Formal definition of quantitative or non-quantitative constraints like	Discovery, composition and interoperabilit

	QoS (Quality of service) requirements like minimum cost and policy requirements like message encryption.	y of Web Services
Execution Semantics	Formal definition of the execution or flow of services in a process or of operations within a service.	Process verification and exception handling*

* Process verification involves verifying the correctness (control and data flow) of a process composition. (Fu et al. 2004) The objective of exception-handling is to identify breakdown points in a Web process and define how to overcome from such breakdowns. (Verma et al. 2005)

Now that we understand why semantics are required in Web service descriptions and what kind of semantics is useful, we can proceed to explore how these semantic annotations are created.

3. CREATING SEMANTIC ANNOTATIONS

With the increasing number of Web services and independent domain models being created, a semi-automatic approach to annotating Web services is very crucial. The fundamental idea behind the association of semantics with Web service elements is to find the most appropriate semantic concept in an ontology for a WSDL element. This is done by matching a WSDL and a domain model schema. For the sake of simplicity let us assume that the domain models have been created using OWL, although they could well be represented in RDF, UML, etc.

Matching a WSDL (basically XML) and OWL schema introduces the problem of matching two heterogeneous models, each with its own expressiveness, capabilities and restrictions. The problem of matching two schemas dates back to the problem of data interoperability in the context of database schemas. The words matching and mapping have often been used interchangeably in the literature. In this chapter, the word schema matching refers to the process of finding semantic correspondences between elements of two schemas and mapping deals with the physical representation of the matches established by schema matching and the rules for transforming elements of one schema to that of the other. For example in Figure 7 that shows a WSDL element and an OWL concept, the result of schema matching is to identify that the POAddress object in the WSDL is semantically equivalent to the Address concept in the ontology. The mapping shown as XQuery (XQuery 1.0: An XML Query Language) and XSLT (XSL Transformations (XSLT)) scripts make the matching operational by specifying rules for transforming elements of one schema to that of the other. Sections 3.1 and 3.2 discuss matching and mapping in the context of Semantic Web services.

3.1 Matching

As far as the problem of schema matching goes, there has been significant work in the database community during 1980s and early 1990s on recognizing the need for data interoperability, schema mapping/merging/transformations, semantic heterogeneity, and use of ontology and description logics for schematic and semantic integration, etc. (e.g., see the discussion in (Sheth 2004)). This was followed by work on schema matching and mapping as part of the Model Management initiative (Model Management). There is ongoing work in the above areas especially in the context of new Web Service technologies and Semantic Web languages (XML, RDF/RDFS, OWL) (Patil et al. 2004, Kalfoglou et al. 2003, Stumme et al. 2001, F Hakimpour et al. 2005).

However, much of the past work in database integration has focused on matching homogeneous models, for example, two database schemas. Any difference in schema representation has been dealt with normalizing the disparate schemas before matching. In the case of matching a WSDL (XML schema) and OWL schema, we are really dealing with two different models. Transforming a less expressive model (XML) to a more expressive model (OWL) would usually require humans to supply additional semantics, while transformation in the other direction can be lossy at best.

Current work in the area of model management (Melnik 2004, Melnik 2005) has focused on developing a generic infrastructure that abstracts operations on models (i.e., schemas) and mappings between models as high level operations which are generic and independent of the data model and application of interest. In the area of Web services, (Patil et al. 2004) addresses the difference in expressiveness between OWL and WSDL (XML) by normalizing both the representations to a common graph format. The result of matching is to establish semantic correspondences which are then represented as annotations. The possible use of machine learning techniques to create metadata for Web services has been explored in ASSAM (Hess et al. 2004a). The annotator component of ASSAM (Hess et al. 2004b) casts the problem of classifying operations and data types in a Web Service as a text classification problems. The tool learns from Web Services with existing semantic annotations and given this training data, semantic labels for unseen Web Services are predicted. A similar attempt at using machine learning techniques is presented in (Oldham et al. 2004).

A semi-automated system for creating annotations on Web Service elements should therefore be able to match a WSDL schema and one or more domain model schemas and return the semantic correspondences with the degree of certainty in the matches. In case of ambiguity, user involvement could help refine the matches produced by the system. Although the need for schema matching is quite obvious (to generate semantic annotations), the need for providing mappings deserves more attention. In Section 3.2, we will discuss the motivation behind mappings, their common representation formats and uses in the context of Web service composition.

3.2 Mapping

As we have seen, semantic annotations on Web service elements facilitate unambiguous service discovery and composition. In the context of service composition, the ordering of services ensures a semantic compatibility between their inputs and outputs but does not necessarily ensure interoperability.

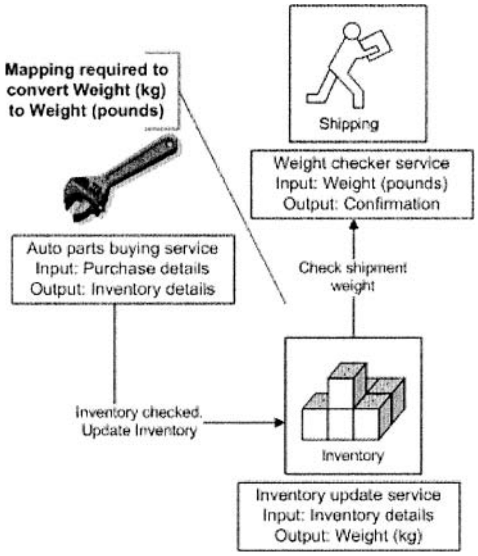


Figure 2-5. A Web process showing the need for mapping between Web service message elements

For example, the Web services shown in Figure 2-5 below make a meaningful process in terms of the semantics of their functionality and the data they exchange, but the format of the messages they exchange is incompatible. The output of the Inventory update service and the input of the Weight checker service are Weight elements and are semantically compatible but differ in their formats (kilograms and pounds), thus making the composition useless at runtime. A mapping between the two elements that converts one message format to another (from Weight in kilograms to Weight in pounds) is required to make this composition operational.

Table 2-3. Possible schematic / data conflicts between xml input/output messages (WSDL-S, Web Service Semantics)

Heterogeneities / Conflicts	**Examples - conflicted elements shown in color**	
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 1*** Student(Id#, Name)	***Web service 2*** Student(SSN, Name) ***Web service 2*** Book (Id#, Name)
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
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
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 1*** TICKET (TicketNo, MovieName)	***Web service 2*** WORKER (Id#, Name) ***Web service 2*** TICKET(FlightNo, Arr. Airport, Dep. Airport)
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)
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)
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)
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, ..., ..)

The generation of mappings like the one in Figure 2-5 is simple and can be automated. More complex mappings however are difficult to automate without human intervention. Table 2-3 illustrates some schema and data conflicts that make the generation of mappings a challenge.

Now that we have recognized the need for such mappings, how would one go about representing and associating these mappings with Web service elements? Clearly, creating mappings between the message elements of two Web services that need to interoperate is not an efficient proposal. Every time a new Web service is created, all existing interoperable Web services would have to create mappings with the new Web service's message elements in the presence of any heterogeneity. An alternative is to create mappings between the Web service element and the domain model or ontology concept with which the Web service element is semantically associated. The ontologies now become a vehicle through which Web services resolve their message level structural or syntactic heterogeneities. Once the mapping is defined and represented, Figure 2-6 shows how two Web services can interoperate using these mappings to ontology concepts. Steps (1), (2) and (3) facilitate message exchange between two communicating Web services. In the first step (1), the WS1 output message is transformed to the OWL concept to which it is mapped (upcast); the OWL concept is then transformed to the WS2 input message (3) (downcast). It is possible that two Web services are not annotated with or mapped to the same ontology. In this case mappings between ontology concepts have to be defined (2). Since mappings are always provided from the Web service element to the ontology concept, generating inverse mappings (to be able to do Step (3) in Figure 2-6) cannot always be automated and requires some user intervention.

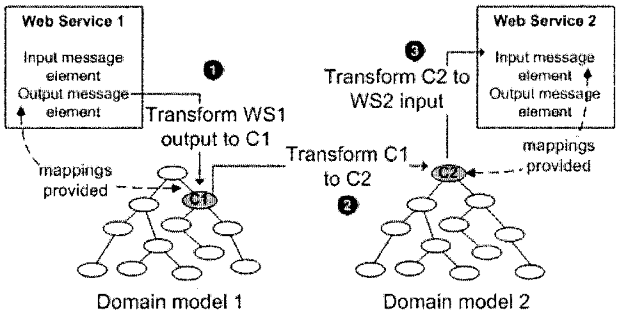
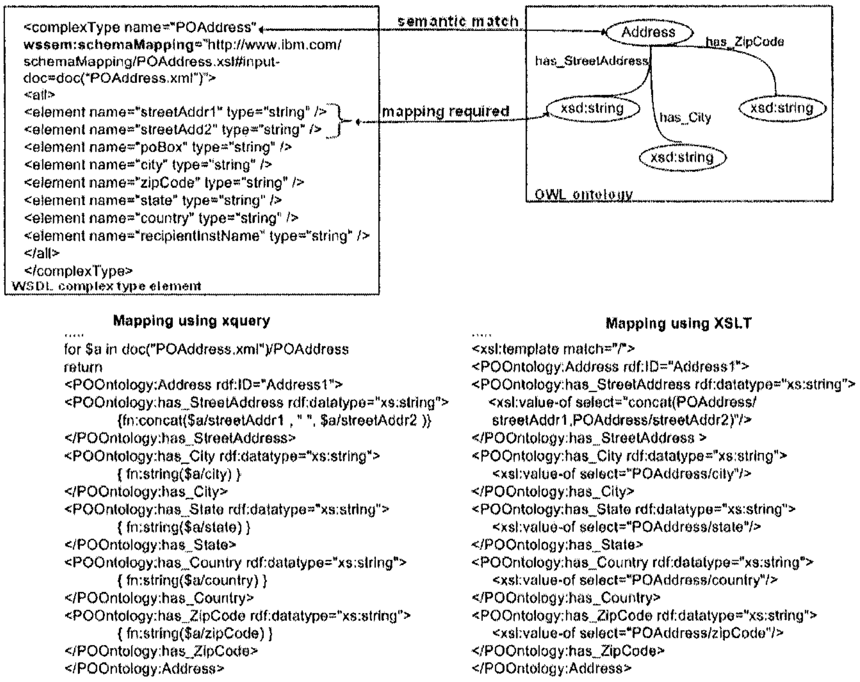


Figure 2-6. Domain models as the vehicle for inter-service communication

In addition to the process of automating the generation of mappings, another research focus has been the representation of the mappings. There have been several approaches to represent mappings in the database literature (Calvanese et al. 2001, Kementsietsidis et al. 2003, A Maedche et al. , Crub'ezy et al. 2003, S.B. Davidson et al. 1995). In the context of Web services, a popular representation for mappings has been the use of XQuery (XQuery 1.0: An XML Query Language) and XSLT (XSL Transformations (XSLT)). Both XQuery and XSLT work on XPATH (XML Linking Language (XLink) Version 1.0) to transform xml objects from one format to another. Figure 2-7 shows an example of a mapping between a WSDL message element and an OWL concept represented using XQuery and XSLT.



```
<complexType name="POAddress"
wssem:schemaMapping="http://www.ibm.com/
schemaMapping/POAddress.xsl#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>
```

Mapping using xquery

```
.....
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_State rdf:datatype="xs:string">
        { fn:string($a/state) }
</POOntology:has_State>
<POOntology:has_Country rdf:datatype="xs:string">
        { fn:string($a/country) }
</POOntology:has_Country>
<POOntology:has_ZipCode rdf:datatype="xs:string">
        { fn:string($a/zipCode) }
</POOntology:has_ZipCode>
</POOntology:Address>
```

Mapping using XSLT

```
.....
<xsl:template match="/">
<POOntology:Address rdf:ID="Address1">
<POOntology:has_StreetAddress rdf:datatype="xs:string">
   <xsl:value-of select="concat(POAddress/
   streetAddr1,POAddress/streetAddr2)"/>
</POOntology:has_StreetAddress >
<POOntology:has_City rdf:datatype="xs:string">
   <xsl:value-of select="POAddress/city"/>
</POOntology:has_City>
<POOntology:has_State rdf:datatype="xs:string">
   <xsl:value-of select="POAddress/state"/>
</POOntology:has_State>
<POOntology:has_Country rdf:datatype="xs:string">
   <xsl:value-of select="POAddress/country"/>
</POOntology:has_Country>
<POOntology:has_ZipCode rdf:datatype="xs:string">
   <xsl:value-of select="POAddress/zipCode"/>
</POOntology:has_ZipCode>
</POOntology:Address>
```

Figure 2-7. Representing mappings using XQuery and XSLT

4. SEMANTIC ANNOTATION OF WEB SERVICES - EFFORTS

The most prominent efforts in the semantic markup of Web services have been OWL-S (OWL-S, OWL-based Web Service Ontology), WSMO (WSMO, Web Services Modeling Ontology) and WSDL-S (WSDL-S, Web Service Semantics). While WSMO and OWL-S define their own rich semantic models for Web services, WSDL-S works in a bottom up fashion by preserving the information already present in the WSDL. In this section, we will briefly discuss these initiatives.

4.1 OWL-S and WSMO

4.1.1 OWL-S

"OWL-S (OWL-S: Semantic Markup for Web Services - White Paper) supplies Web service providers with a core set of markup language constructs for describing the properties and capabilities of their Web services in unambiguous, computer-intepretable form. OWL-S markup of Web services facilitates the automation of Web service tasks including automated Web service discovery, execution, interoperation, composition and execution monitoring. Following the layered approach to markup language development, the current version of OWL-S builds on top of OWL."

OWL-S employs an upper level ontology to describe Web services. The ontology comprises of a service profile *(What does the service provide for prospective clients?)*, service model *(How is it used?)* and service grounding *(How does one interact with it?)*.

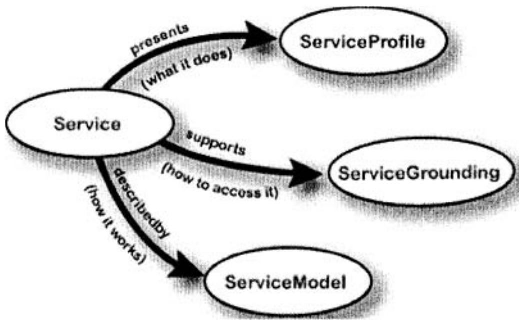


Figure 2-8. Top level of the OWL-S service ontology (OWL-S: Semantic Markup for Web Services - White Paper)

Every instance of a published Web service has an instance of the 'Service' class. The properties of the Service class, 'presents', 'describedBy' and 'supports' point to classes 'ServiceProfile', 'ServiceModel', and 'ServiceGrounding'. "Each instance of a Service will *present* a ServiceProfile description, be *describedBy* a ServiceModel description, and *support* a ServiceGrounding description." The ServiceProfile provides the information needed for an agent to discover a service, while the ServiceModel and ServiceGrounding together provide information for an agent to use the service. Figure 2-8 shows the upper level service ontology in OWL-S.

4.1.2 WSMO

Web Service Modelling Ontology WSMO, also a W3C submission, is a conceptual model for Semantic Web services. It comprises of an ontology of core elements for Semantic Web services, described in a formal description language (WSML) (WSML, Web Services Modeling Language) and also has a execution environment (WSMX) (WSMX, Web Service Execution Environment,). WSMO was derived and based on the Web Service Modelling Framework (WSMF) (D Fensel et al. 2002).

In WSMO, *Ontologies* provide the terminology used by other WSMO elements to describe the relevant aspects of the domains of discourse; *Goals* represent user desires which can be fulfilled by executing a Web service; and *Mediators* describe elements that overcome interoperability problems between different WSMO elements. WSMO considers three levels of mediation - Data Level (to mediate heterogeneous Data Sources), Protocol Level (to mediate heterogeneous Communication Patterns) and Process Level (to mediate heterogeneous Business Processes).

WSMO and OWL-S, both adopt the same view towards having service ontologies to build semantic Web services. OWL-S is based on OWL and represents rules using the Semantic Web Rule Language (SWRL). WSMO has it own family of languages WSML which is based on Description Logics (Description Logics) and Logic programming (Lloyd 1987).

4.2 WSDL-S

WSDL-S, very recently submitted to the W3C, provides a mechanism to annotate the capabilities and requirements of Web services (described using WSDL) with semantic concepts defined in an external domain model. Annotations are achieved using WSDL extensibility elements and attributes. Figure 2-9 shows how semantic annotations are associated with various elements of a WSDL document (including inputs, outputs and functional aspects like operations, preconditions and effects) by referencing the semantic concepts in an external domain semantic model. The domain model can consist of one or more ontologies.

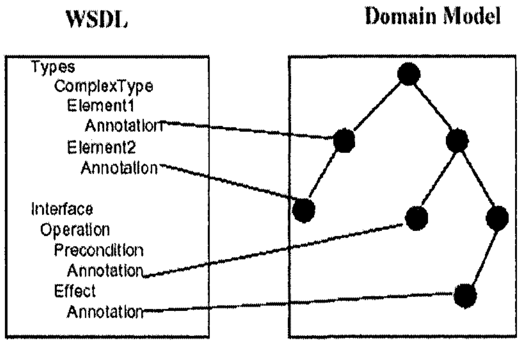


Figure 2-9. Externalized representation and association of semantics to WSDL elements (WSDL-S, Web Service Semantics)

By building on existing Web service standards, something the community is already familiar with, WSDL-S shows good promise of acceptance and quick realization. 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 (not necessarily in OWL as required by OWL-S) or reuse existing domain models. This is a huge advantage since before OWL was popular, quite a few domain models were developed using RDF/S and UML. Table 2-4 below shows the basic extensibility elements and attributes used by WSDL-S and their purpose. The reader should notice that WSDL-S refers to what OWL-S calls the profile model. The OWL-S process model compares with BPEL4WS (Business Process Execution Language for Web Services version 1.1,) and is not a part of the WSDL-S specification.

Figure 2-10 shows an example of a WSDL-S document. In this WSDL-S, "processPurchaseOrder" is an operation whose output message element "processPurchaseOrderResponse" has been annotated using the *modelReference* and *schemaMapping* attributes. Also present are semantic annotations on the preconditions and effects of the operation and a category annotation on the interface element. Associating semantics with these Web service elements enables automation of service discovery, composition and invocation.

Table 2-4. WSDL-S Extensions (Gomadam et al. 2005)

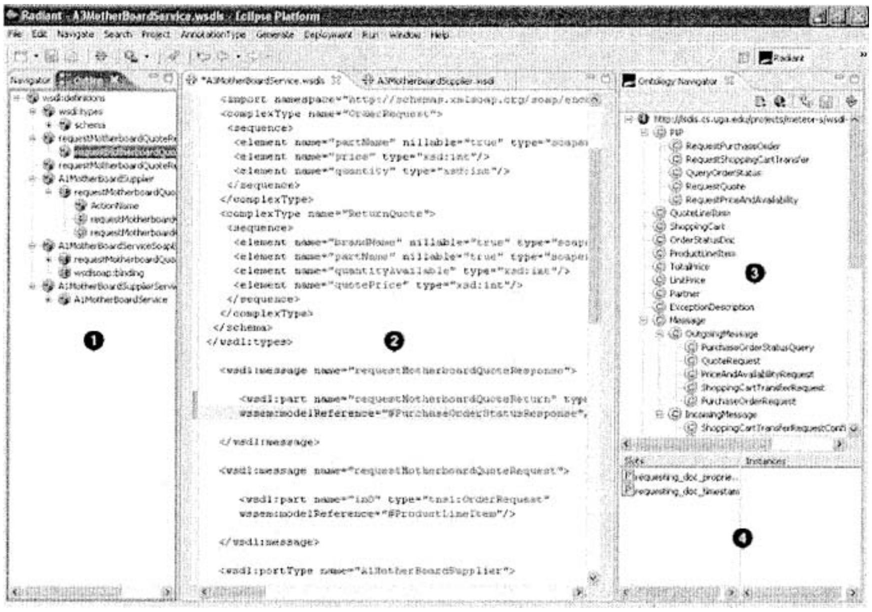
Extension Element / Attribute	**Description**
modelReference (Element: Input and Output Message types)	Semantic annotation of WSDL input and output message types with concepts in a semantic model.
schemaMapping (Element: Input and Output Message types)	Association of structural and syntactic mappings between WSDL message types and concepts in a semantic model.
modelReference (Element: Operation)	Captures the semantics of the functional capabilities of an operation.
pre-conditions (Parent Element: Operation)	Set of semantic statements (or expressions represented using the concepts in a semantic model) that are required to be true before an operation can be successfully invoked
effects (Parent Element: Operation)	Set of semantic statements (or expressions represented using the concepts in a semantic model) that must be true after an operation completes execution.
category (Parent Element: Operation)	Service categorization information that could be used when publishing a service in a Web Services registry such as UDDI.

```
.....
<xs:element name= "processPurchaseOrderResponse" type="xs:string
wssem:modelReference="POOntology#OrderConfirmation"
wssem:schemaMapping="http://lsdis.cs.uga.edu/projects/meteor-s/wsdl-s/examples/
sample.xq"/>
</xs:schema>
</types>
<interface name="PurchaseOrder">
<wssem:category name="Electronics" taxonomyURI="http://www.naics.com/"
    taxonomyCode="443112">
<operation name="processPurchaseOrder" pattern=wsdl:in-out
modelReference = "rosetta:#RequestQuote" >
<input messageLabel = "processPurchaseOrderRequest"
element="tns:processPurchaseOrderRequest"/>
<output messageLabel ="processPurchaseOrderResponse"
element="processPurchaseOrderResponse"/>
<!—Precondition and effect are added as extensible elements on an operation>
<wssem:precondition name="ExistingAcctPrecond"
wssem:modelReference="POOntology#AccountExists">
<wssem:effect name="ItemReservedEffect"
wssem:modelReference="POOntology#ItemReserved"/>
</operation>
</interface>
```

Figure 2-10. Sample WSDL-S document

4.2.1 Radiant - WSDL-S Annotation Tool

The semi-automatic creation of a WSDL-S document depends on the automation of the matching and mapping process discussed in Section 3. Radiant (Gomadam et al. 2005), a manual WSDL-S annotation tool allows users to create WSDL-S files by providing a 'point and click' and 'drag and drop' interface to annotate an existing WSDL file using one or more ontologies. Figure 2-11 shows a snap shot of the tool. The tool offers tree representations of the source WSDL files (1) and OWL files ((3) and (4)) simultaneously to let the user choose the most appropriate semantic match. The WSDL-S file that is generated is shown in (2).



```
<import namespace="http://schemas.xmlsoap.org/soap/enc
<complexType name="OrderRequest">
 <sequence>
  <element name="partName" nillable="true" type="soape
  <element name="price" type="xsd:int"/>
  <element name="quantity" type="xsd:int"/>
 </sequence>
</complexType>
<complexType name="ReturnQuote">
 <sequence>
  <element name="brandName" nillable="true" type="soap
  <element name="partName" nillable="true" type="soape
  <element name="quantityAvailable" type="xsd:int"/>
  <element name="quotePrice" type="xsd:int"/>
 </sequence>
</complexType>
</schema>
</wsdl:types>

<wsdl:message name="requestMotherboardQuoteResponse">

   <wsdl:part name="requestMotherboardQuoteReturn" type
   wssem:modelReference="#PurchaseOrderStatusResponse",

</wsdl:message>

<wsdl:message name="requestMotherboardQuoteRequest">

   <wsdl:part name="in0" type="tns1:OrderRequest"
   wssem:modelReference="#ProductLineItem"/>

</wsdl:message>

<wsdl:portType name="A1MotherBoardSupplier">
```

Figure 2-11. Radiant - WSDL-S Annotation tool (Gomadam et al. 2005)

5. CONCLUSIONS

Creating semantic markup of Web services to realize the vision of Semantic Web services has received a lot of attention in the recent years. A direct offshoot has been the development of agent technologies that can utilize these annotations to support automated Web service discovery, composition and interoperability. For the same reasons we recognize the value add that automated semantic annotation frameworks can bring to the Web service community. This chapter has therefore been an attempt to point the reader to existing work in the areas of semantic annotation of Web resources and Web services in particular. The issues that need to be addressed in the context of annotation of Web services are quite different from traditional Web resource annotation frameworks and therefore deserve attention. Challenges of automating (or reducing human involvement) the matching of heterogeneous schemas, representation and use of mappings for Web services are constantly being addressed.

The interested reader is encouraged to refer to resources mentioned in the Additional Readings section 7 below to gain an in-depth understanding of related topics. A Google Scholar search on topics like Semantic Heterogeneity that introduces the case for matching, Evaluation of metadata quality, Disambiguation etc. are possible resources to look at. Projects such as METEOR-S (METEOR-S: Semantic Web Services and Processes) focus on the use of semantics in the life cycle of Web services. Readers are encouraged to visit the web site of METEOR-S and that of similar projects to stay abreast with the state-of-the-art technology and research.

6. QUESTIONS FOR DISCUSSION

Beginner:
1. Why is there a need for semantic markup of Web resources?
2. What is Entity Disambiguation?
3. What does the semantic markup of Web services offer or enable?
4. What are the four types of semantics that are useful in the life-cycle of a Web process?
5. Define semantic matching and mapping and give an example.
6. What is the fundamental difference between what WSDL-S advocates and the approach used by WSMO or OWL-S?

Intermediate:
1. Why do Semantic Web annotation tools need to disambiguation capability built into them?
2. Discuss how the annotation of Web services is different from annotation of a text document.
3. At what stages of the life-cycle of a Web process are the four semantics used?
4. Why is data integration a problem in Web services and how are ontologies used to facilitate this problem?

Advanced:
1. How can one measure the quality of annotations generated by semantic annotation tools?
2. Why is a semantic match between message elements not sufficient to make a service composition operational?
3. Compare and contrast WSDL-S with OWL-S and WSMO?

7. SUGGESTED ADDITIONAL READING

Topic: Matching, Mapping, Disambiguation

- Semantic Heterogeneity in Global Information Systems - The Role of Metadata, Context and Ontologies (Kashyap et al. 1998)
- Semi-automatic Composition of Web Services using Semantic Descriptions (Sirin et al. 2002)
- Generic Model Management: Concepts and Algorithms (Melnik 2004)
- Reference reconciliation / Disambiguation (Dong and al 2005)

Topic: General
- Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics (Sheth 1998)
- Image Annotation (Hollink et al. 2003) (Wenyin et al. 2001)
- Evaluating the quality of metadata or annotations (Metadata Quality Evaluation)
- A Conceptual Architecture for Semantic Web Enabled Web Services (Bussler et al. 2002)

Projects and initiatives
- METEOR-S (METEOR-S: Semantic Web Services and Processes)
- Semantic Web Services Interest Group (Semantic Web Services Interest Group)
- Semantic tools for Web services (Semantic tools for Web services)
- Semantic Web Services Initiative (SWSI) (Semantic Web Services Initiative)
- Semantic Web Services Home Page (Semantic Web Services Home Page)

8. REFERENCES

A Maedche et al., MAFRA.
Annotation Tools http://annotation.semanticweb.org/tools
Berners-Lee T et al., The Semantic Web. Scientific American. 2001.
Blume M, Automatic Entity Disambiguation: Benefits to NER, Relation Extraction, Link Analysis, and Inference. International Conference on Intelligence Analysis, 2005.
Business Process Execution Language for Web Services version 1.1, http://www-128.ibm.com/developerworks/library/specification/ws-bpel/
Bussler C et al., A Conceptual Architecture for Semantic Web Enabled Web Services. 2002.
Calvanese D et al., Ontology of integration and integration of ontologies. In Description Logic Workshop 2001, 10-19.
Crub´ezy M et al. Ontologies in support of problem solving. Springer, 2003.
D Fensel et al., The Web Service Modeling Framework WSMF. 2002 Electronic Commerce Research and Applications, 1 (2).
Description Logics http://dl.kr.org/

Dill S et al., SemTag and seeker: bootstrapping the semantic web via automated semantic annotation. 2003.

Dong X et al., Reference Reconciliation in Complex Information Spaces. 2005.

Extensible Markup Language (XML) http://www.w3.org/XML/

F Hakimpour et al., Resolution of Semantic Heterogeneity in Database Schema Integration Using Formal Ontologies. 2005 Information Technology and Management.

Fu Xiang et al., Analysis of interacting BPEL web services. WWW, 2004, 621-630.

Goble CA et al., Conceptual Open Hypermedia = The Semantic Web? 2001.

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

Guha R. et al., Tap: Towards a web of data. . http://tap.stanford.edu/.

Hammond et al., Semantic Enhancement Engine: A Modular Document Enhancement Platform for Semantic Applications over Heterogeneous Content. 2002.

Han H et al., Two supervised learning approaches for name disambiguation in author citations. Proceedings of the 4th ACM/IEEE-CS joint conference on Digital libraries, 2004.

Handschuh S et al., Authoring and annotation of web pages in CREAM. 2002.

Heflin J et al., SHOE: a knowledge representation language for Internet applications. 1999.

Hess A et al., Machine Learning for Annotating Semantic Web Services. 2004a.

Hess A et al., ASSAM: A Tool for Semi-Automatically Annotating Semantic Web Services. 2004b.

Hollink L et al., Semantic annotation of image collections. 2003.

Kahan J et al., Annotea: an open RDF infrastructure for shared Web annotations. 2002.

Kalashnikov DV et al., A probabilistic model for entity disambiguation using relationships. SIAM International Conference on Data Mining (SDM), 2005.

Kalfoglou Y. et al., Ontology mapping: the state of the art: The Knowledge Engineering Review. 2003, 18(1). 1--31.

Kashyap V et al., Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context and Ontologies. 1998.

Kementsietsidis A et al., Mapping Data in Peer-to-Peer Systems: Semantics and Algorithmic Issues. SIGMOD, 2003, 325-336.

KIM Knowledge Base http://www.ontotext.com/kim/KBStatistics.pdf

Kogut P et al., AeroDAML: Applying Information Extraction to Generate DAML Annotations from Web Pages. 2001.

Lloyd JW, Foundations of logic programming. 1987.

Melnik S. Model Management: First Steps and Beyond German Database Conference, 2005

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

Metadata Quality Evaluation. SIG CR. http://www.asis.org/Conferences/AM05/abstracts/216.html

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

Model Management http://research.microsoft.com/db/ModelMgt/

Oldham N et al., METEOR-S Web Service Annotation Framework with Machine Learning Classification. Proceedings of the 1st International Workshop on Semantic Web Services and Web Process Composition (SWSWPC'04), In Conjunction with the 2nd International Conference on Web Services (ICWS'04), 2004, 137 - 146.

OWL-S, OWL-based Web Service Ontology. http://www.daml.org/services/owl-s/

OWL-S: Semantic Markup for Web Services - White Paper. http://www.daml.org/services/owl-s/1.0/owl-s.html

Patil A et al., METEOR-S Web service Annotation Framework. The Proceedings of the Thirteenth International World Wide Web Conference, 2004, 553-562.
Popov B et al., KIM - Semantic Annotation Platform. 2003.
PROTON Ontology http://proton.semanticweb.org/
RDF Vocabulary Description Language 1.0: RDF Schema http://www.w3.org/TR/rdf-schema/
Reeve L et al., Survey of Semantic Annotation Platforms. 2005.
Resource Description Framework. http://www.w3.org/RDF/
S.B. Davidson et al., Semantics of Database Transformations: Semantics in Databases. 1995. 55-91.
Semantic Search - The SHOE Search Engine http://www.cs.umd.edu/projects/plus/SHOE/search/
Semantic tools for Web services http://www.alphaworks.ibm.com/tech/wssem
Semantic Web Activity Statement http://www.w3.org/2001/sw/Activity
Semantic Web Services Home Page http://www.daml.org/services/
Semantic Web Services Initiative http://www.swsi.org/
Semantic Web Services Interest Group http://www.w3.org/2002/ws/swsig/
Sheth A., Changing Focus on Interoperability in Information Systems: From System, Syntax, Structure to Semantics. 1998 Interoperating Geographic Information Systems. 5-30.
Sheth A. Early work in database research on schema mapping/merging/ transformation, semantic heterogeneity, and use of ontology and description logics for schematic and semantic integration Dagstuhl Seminar on Semantic Interoperability and Integration, 2004.
Sheth A.P. Semantic Web Process Lifecycle: Role of Semantics in Annotation, Discovery, Composition and Orchestration, Workshop on E-Services and the Semantic Web, 2003.
The SHOE Knowledge Annotator http://www.cs.umd.edu/projects/plus/SHOE/KnowledgeAnnotator.html
Sirin E et al., Semi-automatic Composition of Web Services using Semantic Descriptions. 2002.
Sivashanmugam K. et al., Adding Semantics to Web Services Standards. Proceedings of the 1st International Conference on Web Services, 2003.
Staab S et al., An annotation framework for the semantic web. 2001.
Stumme G et al. FCA-Merge: Bottom-up merging of ontologies 7th Intl. Conf. on Artificial Intelligence, Seattle, WA, 2001.
UDDI Technical White Paper http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf
Universal Description, Discovery and Integration http://www.uddi.org/about.html
Verma Kunal et al. Optimal Adaptation in Autonomic Web Processes with Inter-Service Dependencies LSDIS Lab Technical Report, 2005.
Web Ontology Language, OWL http://www.w3.org/TR/owl-features/
Wenyin L et al., Semi-automatic image annotation. 2001.
WSDL-S, Web Service Semantics http://www.w3.org/Submission/WSDL-S/
WSML, Web Services Modeling Language. http://www.wsmo.org/wsml/
WSMO, Web Services Modeling Ontology. http://www.wsmo.org/
WSMX, Web Service Execution Environment, . http://www.wsmx.org/
XML Linking Language (XLink) Version 1.0 http://www.w3.org/TR/2001/REC-xlink-20010627/
XQuery 1.0: An XML Query Language http://www.w3.org/TR/xquery/
XSL Transformations (XSLT) http://www.w3.org/TR/xslt