Book Chapter, Datenbanken und Informationssysteme, Festschrift zum 60. Geburtstag von Gunter Schlageter, Publication Hagen, October 2003-09-26

Metadata and Semantics for Web Services and Processes

Kaarthik Sivashanmugam, Amit Sheth, John Miller, Kunal Verma,
Rohit Aggarwal, Preeda Rajasekaran
Large Scale Distributed Information Systems (LSDIS) Lab,
University of Georgia, USA.

Abstract

There is much to be gained from adding semantics to Web services. This chapter provides a review of current metadata standards for Web services and explains the need for adding semantics to Web services. It also gives an overview of our current work in the area of creating semantic Web services and semantic Web processes. The METEOR-S project at LSDIS lab, which investigates the role of semantics in the Web process lifecycle, is also discussed..

1. Introduction

Web services promise universal interoperability and integration. The key to achieving this relies on the efficiency of discovering appropriate services and composing them to build complex processes. As the discovery of Web services plays a vital role in selecting a service that fits into a requirement, it becomes increasingly necessary to maintain a profile of the Web service using its metadata. In metadata based Web service discovery, the metadata profile of the services is matched against the requirements to find a Web service that better meets the needs. At present, the metadata-based discovery of Web services is limited to the details available in UDDI [1]. The idea of a UDDI registry is based on the representation of metadata about services to find services, partners, specify interaction/collaboration mechanism, etc. By having a well-defined information model, the details about Web services like the service name, provider name, service/provider categorization etc. are stored in UDDI. UDDI supports keyword based or category based service discovery using its API. The UDDI registry is supposed to open doors for the success of service oriented computing leveraging the power of the Internet. Hence the discovery mechanism supported should scale to the magnitude of the Web by efficiently discovering relevant services among tens and thousands (or millions according to the industry expectations) of the Web services. The present discovery supported by UDDI is inefficient as services retrieved may be inadequate due to low precision (many services you do not want) and low recall (missed the services you really need to consider). Effectively locating relevant services and efficiently performing the search operation in a scalable way is what is required to accelerate the adoption of Web services. To meet this challenge, Web service search engines and automated discovery algorithms need to be developed. The discovery mechanisms supported need to be based on machine processable enriched metadata profile of Web services. The topic of machine processable/interpretable metadata is not a new topic of research. The use of metadata for information integration and knowledge management has been long studied [2]. The concept of metadata has evolved from using a restricted vocabulary for describing an

artefact to using ontologies for metadata enhancement [3]. The Semantic Web [4] initiative has added tremendous potential to the use of metadata in information retrieval. Along the similar lines, the use of semantic metadata for Web services will bring about huge success to the Web services. The concept of Semantic Web services [5,6,7,8] has received much attention both in industry and in academia due to its critical importance. The idea behind Semantic Web Services is to semantically represent metadata of Web services to enable automated discovery, composition and execution.

In this chapter, we discuss

- Different types of metadata
- The metadata available in Web services standards like WSDL and UDDI
- Categorization of the metadata available in these standards
- Need for semantic metadata and how it can be added to these standards
- Use of metadata in Web Process lifecycle

2. Metadata

Metadata can be defined as a set of assertions about things in our domain of discourse. Metadata is a component of data, which describes the data. It is "data about data". Often there is more than that, involving information about data as they is stored or managed, and revealing partial semantics such as intended use (i.e., application) of data. This information can be of broad variety, meeting if not surpassing the variety in the data themselves. They may describe, or be a summary of the information content of the individual databases in an intentional manner. Some metadata may also capture content-independent information like location and time of creation.

Metadata descriptions present two advantages [2]:

- They enable the abstraction of representational details such as the format and organization of data, and capture the information content of the underlying data independent of representational details. This represents the first step in reduction of information overload, as intentional metadata descriptions are in general an order of magnitude smaller than the underlying data.
- They enable representation of domain knowledge describing the information domain to which the underlying data belong. This knowledge may then be used to make inferences about the underlying data. This helps in reducing information overload as the inferences may be used to determine the relevance of the underlying data without accessing the data.

Metadata can be classified based on different criteria. Based on the level of abstraction in which a metadata describes content, the metadata can be classified as follows [9]:

- *Syntactic Metadata* focuses on details of the data source (document) providing little insight into the data. This kind of metadata is useful mainly for categorizing or cataloguing the data source. Examples if syntactic metadata include language of the data source, creation date, title, size, format etc.
- *Structural Metadata* focuses on the structure of the document data, which facilitates data storage, processing and presentation such as navigation, eases

information retrieval, and improves display. E.g. XML schema, the physical structure of the document like page images etc.

- *Semantic Metadata* describes contextually relevant information focusing on domain-specific elements based on ontology, which a user familiar with the domain is likely to know or understand easily. Using semantic metadata, meaningful interpretation of data is possible and interoperability will then be supported at high-level (hence easier to use), providing meaning to the underlying syntax and structure.

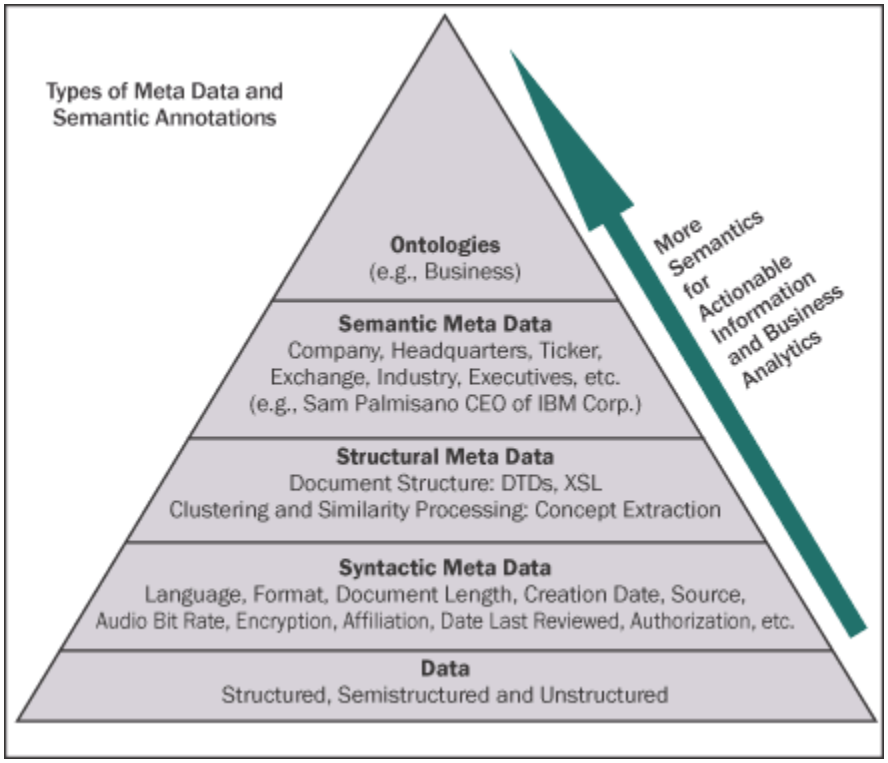

Figure 1: Types of Metadata [9]

The progression of metadata usage from syntax and structure to semantics (Figure 1) increases the control and the insight of data for meaningful interpretation and integration. It is only through semantic metadata that both humans and software can start to associate meaning with data/documents. The use of metadata possibly at multiple levels allows more precise and useful description of artefacts (e.g., data, operations, services, etc.). We can define metadata for metadata in a recursive fashion. This layered structure would make the metadata definitions more semantic and help the interpreting agents to better understand the meaning of the data. Metadata elements can be supplied at multiple and various levels. To satisfy users with entirely different requirements, it is prudent to have multiple levels of metadata (e.g. high-level information, low-level features) or a hierarchical metadata framework, which makes interpretation and information retrieval easier and produces very relevant results during matching.

A common approach to specifying semantics is to use one document in one language for the syntax and another document (could be in another language) for specifying the semantics of the first document (i.e., what its symbols mean, etc.). The second document

formed using the syntax of the second language, for which we could have a third language giving its semantics [10]. Rather than following this recursion indefinitely, at some level (e.g., second or third) we can choose common agreement over explicit definition.

Second level agreements are today being codified as ontologies, which are made Web accessible so that they have wide exposure. An ontology [11, 12] can be defined as a specific vocabulary and relationships used to describe certain aspects of reality, and a set of explicit assumptions regarding the intended meaning of the vocabulary of words. Ontologies consist of definitional aspects such as high-level schemas and assertional aspects such as entities, attributes and relationships between entities, domain vocabulary and factual knowledge all connected in a semantic manner. Ontologies and metadata provide the specific tools to organize and provide a useful description of heterogeneous content. The description incorporates as well as extends an automatic classification-supported approach of organizing content into taxonomy. In addition to the hierarchy of relationship structure of typical taxonomies, ontologies enable cross-node horizontal relationships between entities, thus enabling easy modeling of real-world information requirements.

The third level may be used to provide agreed upon meaning to the second level (e.g., domain ontologies). Currently, work is ongoing to develop upper ontologies that span multiple domains (e.g., multilingual and multi-domain reference ontology, [13]). Only a few domain ontology servers provide libraries with knowledge representation ontologies, common-sense ontologies, upper-level ontologies, generic ontologies that could be reusable across domains, domain ontologies, etc. However, the maturity level of such ontologies is insufficient for the construction of the Semantic Web. The multilingual and multi-domain reference ontology provides formal and detailed knowledge models that allow the vertical intra-operability of systems in specialized domains. It also allows the horizontal interoperability of application in different domains. In the following sections, we discuss different types of metadata available in Web services standards and how the use of ontologies helps in maintaining semantic metadata of Web services.

3. Metadata in WSDL and UDDI standards

The standards such as WSDL [14] and UDDI are used to share the metadata about a web service. Each standard provides metadata about services at a certain level of abstraction. WSDL describes the service using the implementation details and hence it can be considered as a standard to represent the metadata of the invocation details of service. As the purpose of UDDI is to locate WSDL descriptions, it can be thought of as a standard for publishing and discovering metadata of Web services. Considering the details in WSDL and UDDI as metadata of a Web service, the different kind of metadata of Web services available in different standards can be categorized as shown in Table 1.

	WSDL	**UDDI**
Semantic Metadata	Documentation element (Human Readable Semantics), Namespaces, Mapping from WSDL to RDF (R070) [15], Conceptual elements in messages using URL (R120) [15], classification system for operations (UC0032) [16], using RDF [17], Ontologies [18, 19]	Description element (Human Readable Semantics), Ontologies [18, 19, 20]
Structural Metadata	WSDL Schema, Message Definition, Service Reference (UC0027) [16]	UDDI Schemas, Schema Version numbers, Element and Attribute Types and Lengths, tModels etc.
Syntactic Metadata	Interface Description, Transport Protocol, SOAP operation Styles, SOAP use, Message patterns, end point, version of services (R075) [15] and descriptions (R119) [15], Service Metadata (UC0026) [16]	Digital Signatures, Affiliation of Registries, Person, Publisher and owner, Business Entity, Business Service, tModels, Internationalization etc.
Data Format	Semi Structured: XML	Semi Structured: XML
Purpose	Web Service Description for Machine Processability	Representation of data and metadata about Web services

Table 1: Web Services Standards and Metadata Types

4. Semantics for Web Services

In the previous section, we discussed different kinds of metadata available in WSDL and UDDI. Section 2 discussed the power of semantic metadata. In Web services domain, semantics represented by the semantic metadata can be classified into the following types [21], namely

- Functional Semantics
- Data Semantics
- QoS Semantics and
- Execution Semantics

These different types of semantics can be used to represent the capabilities, requirements, effects and execution pattern of a Web service. The semantic Web research focuses to date as focused on the data semantics that helps in semantic tagging of static information available on the Web from all kind of sources. Research on semantic Web services on the other hand is based on the findings and results from the semantic Web research to apply for services that perform some action producing an effect. Unlike information retrieval,

the semantic search is not the only critical aspect with respect to services. In addition to semantic search, service selection, composition and monitoring are to be supported. Automating these steps in Web service usage life cycle is the aim of semantic Web services research. In this section we describe the nature of Web services and the need for different kind of semantics for them.

4.1 Functional Semantics

The power of Web services can be realized only when appropriate services are discovered based on the functional requirements. It has been assumed in several semantic Web service discovery algorithms [22] that the functionality of the services is characterized by their inputs and outputs. Hence these algorithms look for semantic matching between inputs and outputs of the services and the inputs and outputs of the requirement. This kind of semantic matching may not always retrieve an appropriate set of services that satisfy functional requirements. Though semantic matching of inputs and outputs are required, they are not sufficient for discovering relevant services. For example, two services can have the same input/output signature even if they perform entirely different functions. A simple mathematical service that performs addition of two numbers taking the numbers as input and produce the sum as output will have the same semantic signature as that of another service that performs subtraction of two numbers that are provided as input and gives out their difference value as output. Hence matching the semantics of the service signature may result in high recall and low precision. As the precision and recall measures are dependent on the metadata of the artefacts under search, having a good metadata model will help in improving these measures. As a step towards representing the functionality of the service for better discovery and selection, the Web services can be annotated with functional semantics. It can be done by having an ontology called *Functional Ontology* in which each concept/class represents a well-defined functionality. The intended functionality of each service can be represented as annotations using this ontology. The functional ontology is different from the typical ontologies that are being discussed in semantic Web or information retrieval research. The ontologies used in information domain pertain only to semantics of data (similar to nouns in English). The functional ontology on the other hand pertains to the semantics of actions (similar to verbs in English). If a Web service (operation in WSDL files) is annotated using a functional ontology, then service discovery algorithms can exploit these annotations for a better search result set (higher precision and recall).

A functional ontology is a well-defined collection of functionalities such *Travel Ticket Booking, Driving direction servicing, Currency Conversion,* etc. This ontology also has relationship between functionalities like *Travel Ticket Booking* '*supports*' *Destination List Browsing.* This ontology helps in effectively representing functionality of a service. Though ontology is the best way to capture the intricate properties of functionalities, standard vocabularies or taxonomies could be used as well. Taxonomies are typically suited for representing a general domain that Web services belong to. UDDI supports categorizing services into standard taxonomies namely NAICS, UNSCPC and GEO. With the help of these taxonomies, Web services can be respectively categorized based on the industry it belongs to, product that the service handles and geographical location.

4.2 Data Semantics

All the Web services (operations in WSDL file) take a set of inputs and produce a set of outputs. These are represented in the signature of the operations in a WSDL file. However the signature of an operation provides only the syntactic and structural details of the input/output data. These details (like data types, schema of a XML complex type) are used for service invocation. To effectively perform discovery of services, semantics of the input/output data has to be taken into account. Hence, if the data involved in Web service operation is annotated using an ontology, then the added data semantics can be used in matching the semantics of the input/output data of the Web service with the semantics of the input/output data of the requirements. Semantic discovery algorithm proposed in [22] uses the semantics of the operational data.

The ontology used to represent data semantics is a typical ontology that can be encountered in information retrieval research literature. In general, these ontologies will be limited to a domain and hence will have details regarding data in that specific domain. For example, if e-business is considered a domain, then it will have concepts like RFQ (Request for Quote), PO (Purchase Order), PO Acknowledgement. The ontology will also store relationship between these concepts like PO '*is_acknowledged_with*' PO Acknowledgement etc. As mentioned for functional semantics, ontologies can be replaced (with compromises, of course) with standard vocabularies, taxonomies etc. For example, ebXML Core Component Dictionary [23] or RosettaNet Technical Dictionary [24] can be used to represent input/output data in Web services.

4.3 QoS Semantics

Web service selection is a need that is almost as important as service discovery. After discovering Web services whose semantics match the semantics of the requirement, the next step is to select the most suitable service. Each service can have different quality aspect and hence service selection involves locating the service that provides the best quality criteria match. Service selection is also an important activity in web service composition [25]. This demands management of QoS metrics for Web services. Web services in different domains can have different quality aspects. These are called *Domain Independent* QoS metrics. There can be some QoS criteria that can be applied to services in all domains irrespective of their functionality or specialty. These are called *Domain Specific* QoS metrics. Both these kind of QoS metrics need shared semantics for interpreting them as intended by the service provider. This could be achieved by having an ontology (similar to an ontology used for data semantics) that defines the domain specific and domain independent QoS metrics.

For example e-business services can refer to an ontology that describes QoS measures like ISO rating or other certification levels; hospitality services can have an industry specific accreditation or star rating. To share the semantics of these quality metrics, domain specific ontologies could be used. Domain independent QoS metrics can include time, cost, reliability and fidelity [26].

4.4 Execution Semantics

Execution semantics of a Web service encompasses the ideas of message sequence (e.g., request-response, request-response), conversation pattern of Web service execution (peer-to-peer pattern, global controller pattern), flow of actions (sequence, parallel, and loops), preconditions and effects of Web service invocation, etc. Some of these details may not be meant for sharing and some may be, depending on the organization and the application that is exposed as a Web service. In any case, the execution semantics of these services are not the same for all services and hence before executing or invoking a service, the execution semantics or requirements of the service should be verified. Some of the issues and solutions with regard to execution semantics are inherited from traditional workflow technologies. However, the globalization of Web services and processes result in additional issues. In e-commerce, using execution semantics can help in dynamically finding partners that will match not only the functional requirements, but also the operational requirements like long running interactions and complex conversations. Also, a proper model for execution semantics will help in co-ordinating activities in transactions that involve multiple parties.

Traditional formal mathematical models (Process Algebra [27]), concurrency formalisms (Petri Nets [28], state machines [29]) and simulation [30] techniques) can be used to represent execution semantics of Web services. Formal modelling for workflow scheduling and execution are also relevant [31]. With the help of execution semantics process need not be statically bound to component Web services. Instead, based on the functional and data semantics a list of Web services can be short listed, QoS semantics can be used to select the most appropriate service, and execution semantics the service can be bound to a process and used to monitor process execution.

5. Ways of Capturing/Representing Semantics of Web Services

In spite of the success of Web services and the enormous potential of Semantic metadata, there are not many tools available for adding semantic metadata to Web services. However, there have been few initiatives to capture and represent the semantic metadata of services. These initiatives range from creating a new standard to extending Web services standards to bridging Web services standards and metadata standards. In this section we discuss the different approaches of adding semantic metadata to Web services.

5.1. Creating a New Ontology-Based Mark Up Language for Services (DAML-S)

DAML-S [18] is the Web Service ontology being developed to enable automation of services on the semantic Web. Using the DAML-S constructs, properties and capabilities of Web services can be described unambiguously to enable automated service discovery, composition, execution and monitoring. The upper ontology provided in DAML-S helps in representing the knowledge about Web services. By having three different kinds of classes (SERVICEPROFILE, SERVICEMODEL, and SERVICEGROUNDING) in the ontology, DAML-S can be used to advertise the functional ("what the service does") and operational ("how it works" and "how it can be accessed") characteristics of a Web

service. Thus, the DAML-S approach aims to create a new mark-up language to semantically represent the capabilities of a Web service. The functionality is represented using the transformation produced by the service. The inputs are transformed into outputs producing some effects under given preconditions. The quality aspects involved in this transformation process can also be specified (albeit in a very limited form compared to the more comprehensive model of [26]). For better composition, invocation, interoperation, execution and monitoring, the operational characteristics are included in the specification. The operational details include constructs to model processes, model dependencies between different services in a process, etc. To represent the invocation details it provides grounding between an abstract DAML-S description of a service and it concrete WSDL description. However, this monolithic approach of capturing Web service semantics may not be appealing to industry for following reasons:

- All the advantages of using DAML-S are discussed with respect to agent technology. However the present Web services standards are agnostic of agent technologies.
- DAML-S descriptions have to be hand coded.
- Industry practitioners have to learn one more standard.
    - There are already several process modelling languages available
    - It complements and partially competes with existing standards such as WSDL

5.2. Adding Semantics to Existing Standards (METEOR-S)

The METEOR-S [19] project at the Large Scale Distributed Information Systems (LSDIS) Lab focuses on extending the industry adopted and existing Web services standards to add semantics for the purpose of Web service automation. It addresses the semantic issues with regard to Web Services and processes. By extending the existing standards with semantics, METEOR-S project aims to exploit semantics for addressing the following challenges:

- Handling heterogeneity and preserving autonomy of Web services, as they are offered by different providers.
- Achieving scalability in discovery/composition techniques as we move from enterprise level processes to inter-enterprise level to global scale.
- Supporting dynamic and complex nature of Web service interactions.

It is based on the findings and results from the Workflow, Semantic Web, Web Service and Simulation technologies to meet these challenges in a practical and standards based approach. The main goals of the project include:

- Applying semantics in annotation, QoS, discovery, composition and execution of Web services.
- Adding semantics to different layers of Web services conceptual stack [32].
- Using ontologies as underpinning for semantic interoperability of the services.

5.2.1 Semantics at different layers:

The following table illustrates different layers in Web services conceptual stack that are used to add semantics. Table 2 also gives the details on the present scenario, why semantics is needed at that layer and how it can be achieved.

Details	Present scenario	Why semantics?	How?
Description Layer (WSDL)	WSDL descriptions are mainly syntactic (provides operational information and not functional information) Semantic matchmaking of requirements and service descriptions are not possible	Unambiguously understand the functionality of the services and the semantics of the operational data	Using Ontologies to semantically annotate WSDL constructs (conforming to extensibility allowed in WSDL specification version 1.2) to sufficiently explicate the semantics of the – data types used in the service description and – functionality of the service
Publication and Discovery Layers (UDDI)	Suitable for simple searches (like services offered by a provider, services that implement an interface, services that have a common technical fingerprint etc.) – Categories are too broad – Automated service discovery (based on functionality) and selecting the best suited service is not possible	Enable scalable, efficient and dynamic publication and discovery (machine processable / automation)	Use of ontology to categorize registries based on domains and characterize them by maintaining the – properties of each registry [33] – relationships between the registries [33] – Capturing the WSDL annotations in UDDI
Flow Layer (BPEL [34])	Composition of Web services is static. – Dynamic service discovery, run-time binding, analysis and simulation are not supported directly	Design (composition), analysis (verification), validation (simulation) and execution (exception handling) of the process models – To employ mediator architectures for	Using – Functionality /preconditions/effects of the participating services – Knowledge of conversation patterns supported by the service – Formal mathematical models like process algebra, concurrency formalisms like State Machines, Petri nets etc.

		automated composition, control flow and data flow based on requirements – To employ user interface to capture template requirements and generate template based on that	– Simulation techniques

Table 2: METEOR-S Project and Semantics at Different Layers

5.2.2 Adding Semantics to WSDL

One way of adding semantics to Web services is to add semantics to its descriptions, i.e. by annotating WSDL files. WSDL has been accepted as the *de facto* standard for Web service description. As the WSDL descriptions are mainly syntactic, adding semantic annotations will help in representing/understanding the semantics of the syntactic constructs in WSDL. With the help of the added semantics, the functionality of the service, semantics of the input and output data types, semantics of the preconditions and effects can be mentioned. Using these details, a well-suited service can be discovered based on the requirements. Adding semantics annotations to WSDL files can be done in following the two approaches made in subsequent sections.

5.2.2a Annotating WSDL files

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
  xmlns:LSDISOnt="http://lsdis.cs.uga.edu/proj/meteor/METEORS/TravelServiceOntology.daml"
  xmlns:LSDISExt="http://lsdis.cs.uga.edu/proj/meteor/METEORS/WSDLExtension"
  .....
    <complexType name="TravelDetails">
    <sequence>
  ……
  </complexType>
  .....
</wsdl:types>
<wsdl:message name="OperationRequest">
   <wsdl:part name="in0" type="tns1:TravelDetails" LSDISExt:onto-concept="LSDISOnt:TicketInformation"/>
 </wsdl:message>
<wsdl:message name="OperationResponse">
   <wsdl:part name="return" type="tns1:Confirmation" LSDISExt:onto-concept="LSDISOnt:ConfirmationMessage"/>
 </wsdl:message>
<wsdl:portType name="Travel">
   <wsdl:operation name="buyTicket" parameterOrder="in0" LSDISExt:operation-concept="LSDISOnt:TicketBooking">
     <wsdl:input message="intf:OperationRequest" name="buyTicketRequest"/>
     <wsdl:output message="intf:OperationResponse" name="buyTicketResponse"/>
     <LSDISExt:precondition name="ValidCreditCard" LSDISExt:precondition-concept="LSDISOnt:ValidCreditCard"/>
     <LSDISExt:effect name="TicketBooked" LSDISExt:effect-concept="LSDISOnt:CardCharged-TicketBooked"/>
  </wsdl:operation>
    <wsdl:operation name="cancelTicket" parameterOrder="in0" LSDISExt:operation-concept="LSDISOnt:TicketCancellation">
    …..
</wsdl:portType>  .....
```

Figure 2: WSDL File Extended with Semantic Constructs

In this approach, the WSDL files are mapped to ontological constructs to perform annotations. This type of annotation in discussed in [35]. The annotations discussed in this paper conform to the extensibility support in WSDL version 1.2. With the help of a GUI tool, the constructs in WSDL files can be mapped in ontologies thus explicating the semantics of the WSDL constructs. With these semantic annotations, the meaning of input/output data, functionality of the service and QoS metrics and their meanings in a Web service is shared.

5.2.2b Annotating Web Service Source Code (in Java):

WSDL files are usually generated from the source code or using WSDL editors. Since the typical way is to generate it from the source code, adding semantics to WSDL can be simplified by adding annotations at the source code itself. Most of the information present in the WSDL files is a reflection of source code information in addition to other details like the namespace used for the Web Service, the protocol employed, etc.

Annotating the Web service source code helps in

- Automating the generation of annotated WSDL files.
- Easier management of changes in annotations.
- Automatic handling of Complex datatype representation in WSDL files. A mapping between the standard language datatypes and the XML datatypes can be provided and using this information WSDL file can be generated from annotated WS source code.

JSR 175 [36] specifies a metadata facility for the Java programming language (to be included in the next release of Java (J2SE, V.1.5). This feature allows declaratively specifying metadata information for classes, interfaces, methods and fields. With this feature, tools and libraries can access this metadata during run-time. This specification requirement also specified definition rules for namespaces. JSR 181 [37] is built using JSR 175 to specify Web Services Metadata for the Java platform. The aim of these specification requirements is to leverage the core Web services standards providing an easy to use development. With features like this, Java source code can be automatically compiled and deployed. To enable interoperation, tools will be created that provide metadata (WSDL file or Java file with proxy) of the service to other applications. Since automatic WSDL generation is the main application of this feature, in addition to the metadata that is needed for deployment and invocation, we are planning to use it to add semantic metadata. Some suggested Web service source code annotations to add semantic metadata are described in Table 3.

Tags	Explanation	Example
@operation	To represent a method as a web service	@operation bookTicket [operation-concept=LSDISOnt:TicketBooking]
@inputparameter	Information about the input parameters	@inputparameter TravelDetails [onto-concept=LSDISOnt:TicketInformation]
@outpupar	Information about the	@outputparameter Confirmation [onto-

ameter	output parameters	concept=LSDISOnt:ConfirmationMessage]
@precondition	Conditions that must be satisfied before a method gets executed	@precondition [name=ValidCreditCard, precondition-concept =LSDISOnt:ValidCreditCard]
@effect	Changes that must happen during or after the execution of a method.	@postcondition [name=TicketBooked, effect-concept=LSDISOnt:CreditCardCharged-TicketBooked]

Table 3: Annotation Constructs in Java Source Code

Using these constructs Java Web service source code can be annotated as shown in figure 3. This annotated source code can be used to generate a WSDL file shown in figure 2.

```
public class  Travel{
….
@operation  buyTicket [operation-concept=LSDISOnt:TicketBooking]
@inputparameter TravelDetails [onto-concept=LSDISOnt:TicketInformation]
@outputparameter Confirmation [onto-concept=LSDISOnt:ConfirmationMessage]
@precondition [name=ValidCreditCard, precondition-concept=LSDISOnt:ValidCreditCard]
@effect [name=TicketBooked, effect-concept=LSDISOnt: CreditCardCharged-TicketBooked]

Confirmation  buyTicket ( TravelDetails tDetails ) { …}
..
}
```

Figure 3: Example Annotated Java Source Code

5.2.3 Adding Semantics to UDDI

As explained in the previous section, adding semantics to a WSDL file helps in understanding the semantics of WSDL constructs. However, to use semantics in discovery, these semantic constructs have to be registered in UDDI. Sivashanmugam et al [35] utilize the UDDI data structure to store semantic details of a service. In this approach, tModels are used to store the semantics in UDDI. *tModels* are metadata constructs in UDDI data structure that provide the ability to describe compliance with a specification, a concept or a shared understanding. They have various uses in a UDDI registry. Commonly agreed specifications or taxonomies can be registered with UDDI as *tModels*. They can also be used to associate entities with individual nodes in taxonomies. When a *tModel* is registered with UDDI registry, it is assigned a unique key, which can be used by entities to refer to it. To categorize entities in UDDI, *tModels* are used in relation with *CategoryBags*, which are data structures that allow entities to be categorized according to one or more *tModels*. Using the new grouping construct *keyedReferenceGroups* in UDDI version 3 specifications, categorization using *tModels* can be grouped. We have used the *keyedReferenceGroup,* along with *tModels* to group operations with their inputs and outputs. As shown in Figure 4, two *keyedReferenceGroups* can be created for the WSDL file in figure 2 to represent two operations, *buyTicket* and c*ancelTicket* along with their inputs and outputs. Each keyed reference has a *keyValue, which represents an ontological concept,* and a *tModelKey*, which represents the ontology in UDDI. Preconditions and effects need similar technique.

Each operation along with its inputs, outputs, preconditions and effects are grouped using a *tModel* into *keyedReferenceGroups*. This grouping in turn can be used during Web service discovery.

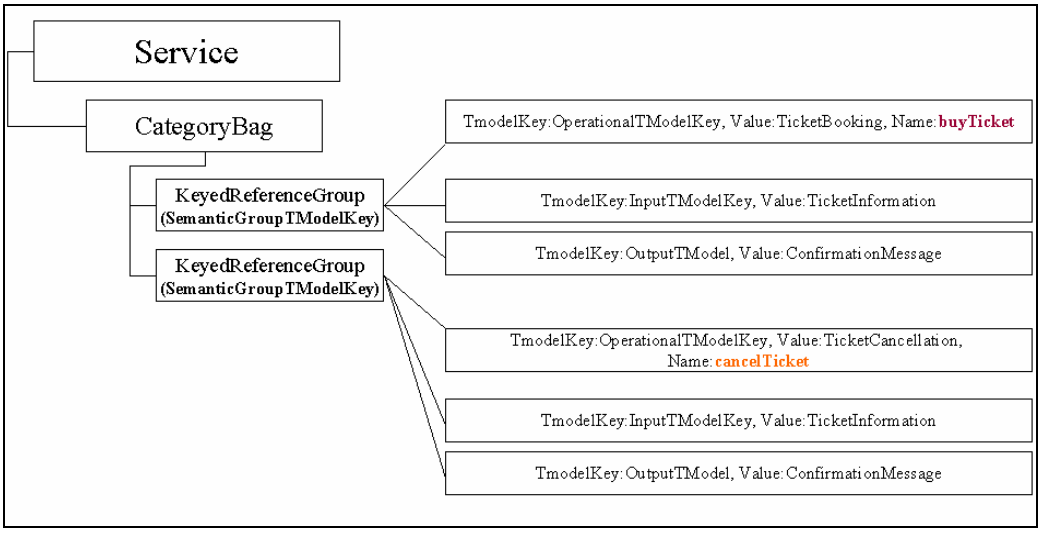

Figure 4: UDDI representation of the WSDL Semantic Annotations

5.3. Other approaches

Apart from the other approaches discussed in sections 5.1 and 5.2, there could be other ways to add semantics to the Web services standards. In this section we discuss two such approaches.

5.3.1 Using RDF

Ogbuji [17] discusses using RDF [38] in conjunction with other standards like WSDL, UDDI and ebXML. It describes how RDF can be used to describe a Web service. It argues that metadata management can be made effective using RDF. This includes representing WSDL in RDF syntax. The advantage of this approach is that it is compatible with the existing RDF based systems (search engines, classification systems), which can be used for Web services domain.

5.3.2 Automatic Generation of Metadata

Heß, and Kushmerick [39] discuss using machine learning and clustering techniques to attach semantic metadata to Web forms and services. It assumes three levels of metadata a taxonomy representing nature of the services, a domain taxonomy representing a collection of functionalities and a datatype taxonomy representing a collection of semantic category of data. Using a clustering technique, the web services are categorized into the classifications based on the details available in WSDL.

6. Web Process Lifecycle and Applications of Semantics at different stages

In order to fully harness the power of Web services, their functionality must be combined to create processes. Semantics plays an important role in all stages of Web process lifecycle. In the following sections we will briefly describe the various stages and also describe the application of semantics to the Web process lifecycle [21].

The main stages in the Web process lifecycle are development of Web services, publication/discovery of Web services, composition of Web processes and execution of Web processes (Figure 5). All these stages are equally important to the Web process lifecycle.

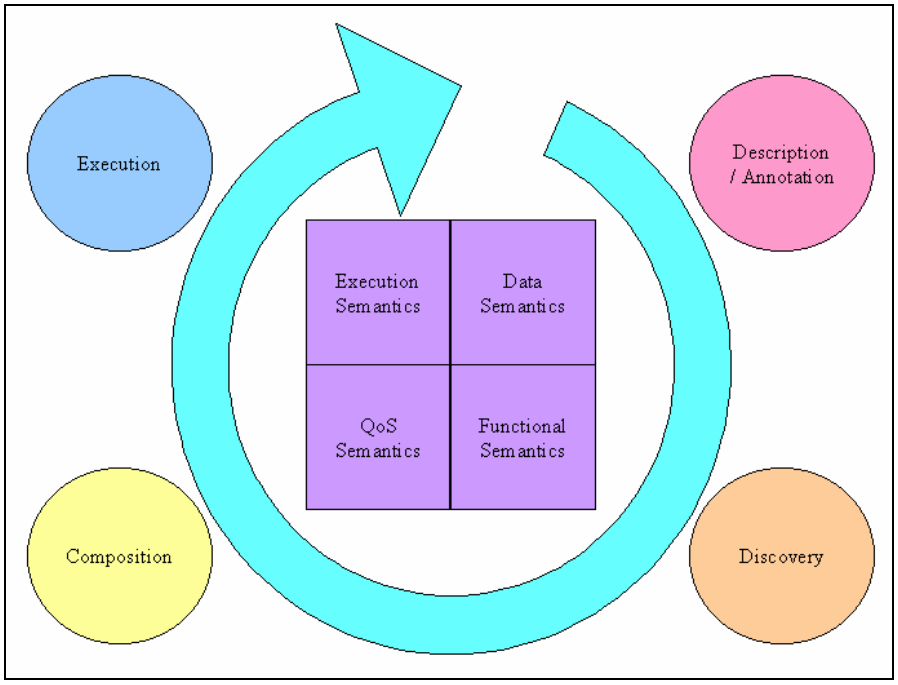

Figure 5: Semantics and Web Process Lifecycle

6.1 Development of Web Services.

Many tools are available to create Web services. Primarily programs written in Java or any object oriented language can be made into Web services In technical terms any program that can communicate with other remote entities using SOAP [40] can be called a Web service. Since development of Web services is the first stage in the creation of Web services, it is very important to use semantics at this stage. During Web service development data, functional and QoS semantics of the service needs to be specified.

6.2 Discovery and Binding of Web Services

After the service is developed, it has to be published to enable discovery. This stage is the process of discovering appropriate Web services and binding them to the Web processes. The data, functional and QoS semantics of the Web services added in the first stage have

to be published to enable finding the desired Web services. Typically, in order to automate the binding process, the Web process is annotated with a semantic representation of the required functionality of the Web services. The binding may be done at design time (static binding) or at runtime (dynamic binding) based on the system requirements. The idea is to create an algorithm, which has three main dimensions for searching Web services. Web services are searched based on functional, data and QoS parameters represented by ontologies. The service that matches requirements most closely is bound to the process. A list of other services, which match the requirements, is maintained. A service may be chosen later in case of failure or breach of contract.

6.3 Composition of Web Processes

This stage involves creating a representation of Web processes. Many languages like BPEL4WS, BPML and WSCI have been suggested for this purpose. The languages provide constructs for representing complex patterns of Web service compositions. While composing a process, all four kinds of semantics have to be taken into account. The process designer should consider the functionality of the participating services (functional semantics), data that is passed between these services (data semantics), the quality of these services, the quality of the process as a whole (QoS semantics) and the execution pattern of these services, the pattern of the entire process (Execution semantics). The power of Web services can be realized only when they are efficiently composed into Web process. Since Web process composition involves all kind of semantics, it may be understood that semantics play a critical role in the success of Web services and in process composition.

6.4 Execution of Web processes

QoS and execution semantics play a very important role in the execution of Web processes. After process composition, the functionality of the process and its execution pattern can be simulated or verified. This will involve using execution semantics. During execution, component services in the process can be replaced by other services providing the same functionality in case of failure or breach of contract. That is, in case a service does not provide the promised QoS, it can be replaced by another service with the same functionality and higher QoS.

7. Summary

The significance of metadata has increased with the emergence of Internet. The need for effective discovery and retrieval of Web resources has always driven researchers to strive for better metadata description. The pace in which Web services are being adopted by the industry demands a proper metadata model for locating Web services. This chapter has concentrated on a semantic approach for creating and using metadata for Web services. We also briefly described how semantics can be applied to the complete lifecycle of Web processes. The use of semantics will help us achieve high scaling and ubiquitous distributed computing solutions using Web services.

8. References

[1] Universal Description, Discovery and Integration of Web Services, http://www.uddi.org/

[2] Sheth, A.: Changing Focus on Interoperability in Information Systems: from System, Syntax, Structure to Semantics. In: Goodchild M.F., Egenhofer, M.J., Fegeas, R., Koffman, C.A. (eds.): Interoperating Geographic Information Systems. Kluwer Academic Publishers, Boston (1999) 5-29

[3] Hammond B., Sheth A. and Kochut K., Semantic Enhancement Engine: A Modular Document Enhancement Platform for Semantic Applications over Heterogeneous Content, in Real World Semantic Web Applications, V. Kashyap and L. Shklar, Eds., IOS Press, ISBN 1-58603-306-9, pp. 29-49. December 2002

[4] W3C Semantic Web, http://www.w3.org/2001/sw/

[5] Ankolenkar A., Burstein M., Hobbs J. R., Lassila O., Martin D. L., McDermott D., McIlraith S. A., Narayanan S., Paolucci M., Payne T. R. and Sycara K., "DAML-S: Web Service Description for the Semantic Web", The First International Semantic Web Conference (ISWC), Sardinia (Italy), June, 2002.

[6] Sheth A. and Meersman R., Amicalola Final Report: SIGMOD Record Special Issue on Semantic Web, Database Management and Information Systems, December 2002

[7] Handschuh S., Sollazzo T., Staab S., Frank M., and Stojanovic N., Semantic Web Service Architecture - Evolving Web Service Standards toward the Semantic Web. The 15th International FLAIRS Conference, Special Track on Semantic Web, Florida, May 14-16, 2002.

[8] Fensel D. and Bussler C., The Web Service Modeling Framework WSMF, http://informatik.uibk.ac.at/users/c70385/wese/wsmf.paper.pdf

[9] Sheth A., Semantic Meta Data for Enterprise Information Integration, DM Review, Vol. 13, No. 7, July 2003, pp. 52-54.

[10] Rapaport, W. J., "Understanding Understanding: Syntactic Semantics and Computational Cognition", in James E. Tomberlin (ed.), *Philosophical Perspectives, Vol. 9: AI, Connectionism, and Philosophical Psychology* (Atascadero, CA: Ridgeview): 49-88. 1995.

[11] Gruber T., The role of common ontology in achieving sharable, reusable knowledge bases. In J. A. Allen, R. Fikes, and E. Sandewall, editors, , San Mateo, CA, 1991. Morgan Kaufman

Book Chapter, Datenbanken und Informationssysteme, Festschrift zum 60. Geburtstag von Gunter Schlageter, Publication Hagen, October 2003-09-26

[12] Guarino N 1998 Formal ontology and information systems. *Proceedings of the 1st International Conference on Formal Ontology in Information Systems [FOIS'98], Torino*: 3-15

[13] Gómez-Pérez A., A Proposal of Infrastructural Needs on the Framework of the Semantic Web for Ontology Construction and Use, http://www.semanticweb.org/SWWS/program/position/soi-gomez-perez.pdf

[14] Christensen E., Curbera F., Meredith G., Weerawarana S., Web Services Description Language (WSDL) 1.1, W3C Note 15 March 2001.

[15] Web Services Description Requirements, W3C Working Draft 28 October 2002 http://www.w3.org/TR/ws-desc-reqs

[16] Web Service Description Usage Scenarios, W3C Working Draft 4 June 2002 http://www.w3.org/TR/ws-desc-usecases/

[17] Ogbuji, U. 2000. Supercharging WSDL with RDF Managing structured Web service metadata, http://www-106.ibm.com/developerworks/library/ws-rdf/?dwzone=ws

[18] DAML Services, http://www.daml.org/services

[19] METEOR-S: Semantic Web Services and Processes Applying Semantics in Annotation, Quality of Service, Discovery, Composition, Execution http://lsdis.cs.uga.edu/proj/meteor/swp.htm

[20] Dogac A., Cingil I., Laleci G., Kabak Y., Improving the Functionality of UDDI Registries through Web Service Semantics, 3rd VLDB Workshop on Technologies for E-Services (TES-02), Hong Kong, China, August 23-24, 2002

[21] Sheth S., Semantic Web Process Lifecycle: Role of Semantics in Annotation, Discovery, Composition and Orchestration. http://lsdis.cs.uga.edu/lib/presentations/WWW2003-ESSW-invitedTalk-Sheth.pdf

[22] Paolucci, M. and Kawamura, T. and Payne, T.R. and Sycara, K. (2002) Importing the Semantic Web in UDDI. In Bussler, C. and Hull, R. and McIlraith, S. and Orlowska, M.E. and Pernici, B. and Yang, J., Eds. Proceedings Web Services, E-Business and Semantic Web Workshop, CAiSE 2002., pages 225-236, Toronto, Canada.

[23] Core Component Dictionary, ebXML Core Components, 10 May 2001. Version 1.04 www.ebxml.org/specs/ccDICT.pdf

[24] RosettaNet Dictionary, http://xml.coverpages.org/RosTechnical-Dictionary.html

[25] Cardoso J., Sheth A., Semantic e-Workflow Composition, Journal of Intelligent Information Systems.

Book Chapter, Datenbanken und Informationssysteme, Festschrift zum 60. Geburtstag von Gunter Schlageter, Publication Hagen, October 2003-09-26

[26] Cardoso, J., Miller, J., Sheth, A. and Arnold, J. "Modeling Quality of Service for Workflows and Web Service Processes." Technical Report, LSDIS Lab, Computer Science, the University of Georgia, May 2002.

[27] Bergstra J. A., Ponse A., and Smolka S. A., Handbook of Process Algebra., Editors. Elsevier, ISBN: 0-444-82830-3, 2001.

[28] Aalst W. M. P., The Application of Petri Nets to Workflow Management, The Journal of Circuits, Systems and Computers. , 8(1): 21-66, 1998.

[29] Hopcroft J. E. and Ullman J. D.. *Introduction to Automata Theory Languages, and Computation*. Addison-Wesley Publishing Company, Reading, Mass., 1979.

[30] Bosilj V., Stemberger M. and Jaklic J., "Simulation Modelling Toward E-Business Models Development", International Journal of Simulation Systems, Science & Technology, Special Issue on: Business Process Modelling, Vol. 2, No. 2, 16-29. (2001).

[31] Attie, P., Singh, M.,. Sheth, A., and Rusinkiewicz, M. "Specifying and Executing Intertask Dependencies," in Proceedings of the 19th Intl. Conference on Very Large Data Bases, August 1993, pp.134-145.

[32] Advancing the Web services stack, http://www-106.ibm.com/developerworks/webservices/library/ws-wsa/

[33] Verma, K., Sivashanmugam, K., Sheth, A., Patil, A., Oundhakar, S. and Miller, J. METEOR–S WSDI: A Scalable Infrastructure of Registries for Semantic Publication and Discovery of Web Services, Journal of Information Technology and Management (to appear, 2003).

[34] Business Process Execution Language for Web Services, Version 1.1 ftp://www6.software.ibm.com/software/developer/library/ws-bpel11.

[35] Sivashanmugam K., Verma K., Sheth A., Miller J., Adding Semantics to Web Services Standards, International Conference on Web Services (ICWS'03), 2003.

[36] Metadata Facility for Java Programming Language, http://www.jcp.org/en/jsr/detail?id=175

[37] Web Services Metadata for the Java Platform, http://www.jcp.org/en/jsr/detail?id=181

[38] Resource Description Framework, http://www.w3.org/RDF/

[39] Heß, A. & Kushmerick, N., Learning to attach semantic metadata to Web Services. *ISWC-2003*.

[40] Simple Object Access Protocol 1.1, http://www.w3.org/TR/SOAP/