Adding Semantics to Web Services Standards

Kaarthik Sivashanmugam, Kunal Verma, Amit Sheth, John Miller
Large Scale Distributed Information Systems (LSDIS) Lab
Department of Computer Science, University of Georgia
Athens, GA 30602

Abstract

With the increasing growth in popularity of Web services, discovery of relevant Web services becomes a significant challenge. One approach is to develop semantic Web services where by the Web services are annotated based on shared ontologies, and use these annotations for semantics-based discovery of relevant Web services. We discuss one such approach that involves adding semantics to WSDL using DAML+OIL ontologies. Our approach also uses UDDI to store these semantic annotations and search for Web services based on them. We compare our approach with another initiative to add semantics to support Web service discovery, and show how our approach may fit current standards-based industry approach better.

Keywords: Semantic Annotation of Web service, Semantic Web service discovery, Semantic Web services, Ontologies, Semantic extensions to WSDL, adding semantics to UDDI

1. Introduction

"Web services are modular, self-describing, self-contained applications that are accessible over the Internet" [1]. They have been identified as the technology for business process execution and application integration. Given the dynamic environment in e-businesses, the power of being able to find Web services on the fly to create business processes is highly desirable. A key step in achieving this capability is the automated discovery of Web services. Currently, the industry standards for Web services are Web Services Description Language [2] and Universal Description Discovery and Integration [3] specifications. Web services are described using WSDL definitions and advertised in UDDI registries. The current discovery mechanism supported by UDDI is not powerful enough for automated discovery. The main inhibitor is the lack of semantics in the discovery process and the fact that UDDI does not use information in the service descriptions during discovery. This makes UDDI less effective, even though it provides an interface for keyword and taxonomy based searching. The key to semantic discovery of Web services is having semantics in the description itself (i.e., "formally self described" [4] and machine processable) and then using semantic matching algorithms to find the required services.

Ontologies have been identified as the basis for semantic annotation that can be used for discovery. Ontologies are the basis for shared conceptualization of a domain [5], and comprise of concepts with their relationships and properties. Use of ontologies to provide underpinning for information sharing and semantic interoperability has been long realized [6, 7, 8]. By mapping concepts in a Web resource (whether data or Web service) to ontological concepts, users can explicitly define the semantics of that resource in that domain. An approach for semantic Web service discovery is to have the ability to construct queries using ontological concepts in a domain. This in turn requires mapping concepts in Web service descriptions to ontological concepts. By having both the description and query explicitly declare their semantics, the results will be more relevant than keyword matching based information retrieval. Our approach of adding semantics in this paper uses ontologies. However we could potentially use enumerated vocabulary also.

There have been a number of efforts to add semantics to the discovery process. An early work in this area has been the creation of DAML-S [9], which uses a DAML+OIL based ontology for describing Web services. While DAML-S provides the expressiveness required for automated discovery, it does not have constructs to represent communication level details of Web services. The latest draft release [10] of DAML-S uses WSDL in conjunction with DAML-S for Web service descriptions. In this paper, we explore the possibilities of adding semantics to WSDL and UDDI to achieve sufficient expressiveness to automate the discovery process. Our approach involves relating concepts in WSDL to DAML+OIL ontologies in Web service description and then providing an interface to UDDI that allows querying based on ontological concepts. WSDL has been accepted as the industry standard for Web service description. If extending it without adding significant complexity could provide the same functionality as DAML-S, our approach may be more attractive to industry and practitioners compared to that of migrating from WSDL to DAML-S which entails the use of a more complex and non-standard framework. Since our approach is backward compatible with existing WSDL standards, service providers also have an option to describe and publish their services with or without semantics. We further provide a matching algorithm to use this semantic information for Web service discovery that considers not only inputs and outputs, but also functional specification of operations and effects. Matchmaking with DAML-S as described in [11] does not consider operations. Compared to semantic annotation of data, Web services add the new dimension of operation. Hence we consider this component of our work to be of critical importance. The work discussed in this paper forms a part of the METEOR-S project, which seeks to address the entire lifecycle of Semantic Web Process, involving semantic specification, annotation, discovery, composition and orchestration of Web services.

In this paper we first outline our approach for adding semantics in WSDL and UDDI. Then, we discuss our semantic discovery algorithm. Thereafter, we compare our approach with DAML-S for adding semantics to Web services and using it for discovery. The rest of the paper is organized as follows. Section 2 outlines the related work. Our approach of adding semantics to WSDL and UDDI are discussed in Sections 3 and 4, respectively. Semantic Web service discovery is discussed in Section 5. Section 6 provides a comparison of our approach with DAML-S approach. Finally in Section 7, we present conclusions and future work.

2. Adding Semantics in WSDL

Currently Web services are described using WSDL descriptions, which provide operational information. Although WSDL descriptions do not contain (or at least explicate) semantic description, they do specify the structure of message components using XML schema constructs. We suggest adding semantics to WSDL using extensibility in elements and attributes supported by WSDL specification version1.2. Using this extensibility we relate existing and extended WSDL constructs to DAML+OIL ontologies. The use of ontologies allows to represent Web service descriptions in a machine-interpretable form like DAML-S. These extensions are similar to the extensions suggested for *ServiceGrounding* in DAML-S.

2.1. Mapping Operations to Ontological Concepts

Recent tools like Web Services Invocation Framework [12] allow invoking Web services if the locations of the WSDL file and the name of the operation are known. So service discovery involves not only locating the WSDL description, but also the relevant operation to invoke. Each WSDL description may have a number of operations with different functionalities. The WSDL file[1] shown in figure 3 represents a sample Web service and has operations for both booking and canceling flight tickets. In order to add semantics and to find relevant operations, these operations should be mapped to concepts in appropriate DAML+OIL ontologies that depict functionality of operations. In figure 3, the operations *buyTicket* and c*ancelTicket* are mapped to ontological concepts *TicketBooking* and *TicketCancellation*, using the attribute operation-concept, respectively. This allows users to search for operations based on ontological concepts. An approach to store the mapping between operation names and ontological concepts in UDDI is discussed in section 4.

2.2. Mapping Message Parts to Ontological Concepts

Message parts, which are input and output parameters of operations, are defined in WSDL files using XML schema constructs. XML schemas could be used as shared definitions of concepts. Since service providers typically embed the schema definitions as inline elements along with service descriptions, it becomes difficult to share and reuse them. Ontologies, which are more expressive and meant for sharing definitions, can be used to annotate the message parts in WSDL. Using ontologies not only brings user requirements and service advertisements to common conceptual space, but also helps to apply reasoning mechanism to find a better match. Hence by using DAML+OIL ontologies in WSDL, the semantics implied by these structures in service descriptions, which are known only to the writer of the description (provider of Web service), can be made explicit. In figure 3, the WSDL constructs input *TravelDetails* and output *Confirmation* are mapped[2] to ontological concepts *TicketInformation* and *ConfirmationMessage*, respectively.

2.3. Adding New Tags for Preconditions and Effects

[1] The names spaces LSDISOnt and LSDISExtension respectively contain TravelServices ontology and the extended WSDL schema used in the examples

[2] In a related work, we are investigating semantic heterogeneity between schema constructs in WSDL and ontological concepts during mapping of message parts to ontology [13]

Each operation may have a number of preconditions and effects. The preconditions may be some logical conditions, which must be true for executing the operation. Effects are changes in the world after the execution of the operation. We propose adding precondition and effect elements as children of the operation element in WSDL. Figure 3 shows the added preconditions and effects to each of the operations in the WSDL description. The operation *buyTicket* has the precondition and effect mapped to ontological[3] concepts *ValidCreditCard* and *CardCharged-TicketBooked-ReadyForPickUp* respectively.

We believe that preconditions and effects are important for Web service selection. After matching services based on operations, inputs and outputs, preconditions and effects could be used to select the most relevant service. It is possible for a number of operations to have the same functionality, as well as, the same inputs and outputs, but different effects. For example, there could be an operation called *bookTicketAndSend*, with the same functionality, inputs and outputs as *buyTicket* in figure 3, but with a different effect called *"CardCharged-TicketBooked-Sent"*. Upon execution, *bookTicketAndSend* sends tickets to the user of the service rather than making them ready for pickup. In this case, depending on the requirements of the user, the most relevant operation can be chosen.

3. Adding Semantics in UDDI

We provide semantic discovery using UDDI by doing the following two tasks. Firstly, we store the semantic annotation of Web services mentioned in section 3 in the existing structures of UDDI. Secondly, we provide an interface to construct queries that use that semantic annotation. This approach is similar to the one suggested by [14], which maps DAML-S to UDDI structures, but is consistent with the use of industry standard WSDL rather than requiring DAML-S.

UDDI only supports a limited form of semantics using *tModels*, which are used to characterize and categorize businesses and their services. During a Web service publication, ontological concepts representing operations, and their message parts, preconditions, effects of the WSDL descriptions of the Web service are stored using the UDDI structures, *tModels* and *CategoryBags*. *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 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 propose using the *keyedReferenceGroup,* along with *tModels* to group operations with their inputs and outputs.

To represent the semantic information in UDDI, we have created four *tModels* in that registry. The first *tModel* represents the ontology of concepts representing functionality of operations in a relevant domain, the second and third represent the ontologies of input and output concepts respectively. Finally, the fourth *tModel* represents the grouping of each operation with its inputs and outputs. These *tModels* are linked with the respective ontologies using *overviewURL* tag of these *tModels*. All the *tModels* could as well be linked to a single comprehensive ontology. As shown in figure 1, two *keyedReferenceGroups* can be created for the WSDL file in figure 3 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 itself. For example, the *tModel* OPERATION_CONCEPTS is used to store the mapping between an WSDL operation and a concept in ontology. It contains the name of the operation as *keyName* and the ontological concept it is mapped to as *keyValue*. Similarly the inputs and outputs of each operation are mapped using INPUT_CONCEPTS and OUTPUT_CONCEPTS *tModels*, respectively. Preconditions and effects need similar technique (not shown in figure 1). Each operation along with its inputs, outputs, preconditions and effects are grouped using MAPPINGGROUP *tModel* into *keyedReferenceGroups*.

4. Semantic Web Service Discovery

Semantic annotations added in WSDL and in UDDI are aimed at improving discovery and composition of services. In this section we briefly describe our mechanism for template based ontology enabled discovery. Figure 2 shows the conceptual process of mapping WSDL constructs given in figure 3, to the nodes in a domain specific ontology. This mapping is then

[3] A discussion of creating an ontology depicting preconditions and effects are beyond the scope of this paper.

stored in UDDI during Web service publication. As shown in the figure 2, the operations *buyTicket* and *cancelTicket* are mapped to the nodes *TicketBooking* and *TicketCancellation*, respectively, the input concept *TravelDetails* and output concept *Confirmation* in WSDL file are mapped to the *TicketInformation* node and *ConfirmationMessage* in the *TravelServices* ontology, respectively.

We have developed a three-phase algorithm for semantic Web service discovery that requires the users to enter Web service requirements as templates constructed using ontological concepts. In the first phase, the algorithm matches Web services (operations in different WSDL files) based on the functionality[4] they provide. In the second phase, the result set from the first phase is ranked on the basis of semantic similarity [15] between the input and output concepts of the selected operations and the input and output concepts of the template, respectively. The optional third phase involves ranking based on the semantic similarity between the precondition and effect concepts of the selected operations and preconditions and effect concepts of the template. Figure 2 shows the creation of a template[5] using ontological nodes for semantic discovery of services. The template has the operation concept *TicketBooking*, the input concept *TicketInformation* and the output concept *ConfirmationMessage*. The template created by the user is converted to a UDDI query by our interface [16]. This template would map to an UDDI query which first searches for all Web services categorized using a *keyedReferenceGroup*[6] which has the *TicketBooking* mapped to the operation *tModel*. The result set is then ranked based on the semantic similarity [15] between the input concepts of the returned Web services to the input concepts (*TicketInformation*) of the template and the output concepts of the returned Web services to the output concepts (*ConfirmationMessage*) of the template.

5. Related Work

DAML-S [9] is based on DAML+OIL and it provides an ontology markup language expressive enough to semantically represent capabilities and properties of Web services. Its goals are to achieve automatic Web service discovery, invocation, composition and execution monitoring. DAML-S has an upper ontology, which characterizes Web services using three types of knowledge about the services - *ServiceProfile*, *ServiceModel* and *ServiceGrounding*. *ServiceProfile* is used to describe what a Web service does, *ServiceModel* describes how it works and *ServiceGrounding* is used to specify how to access it. Paolucci et al [11] presents a mapping engine to match service advertisements with requests. It provides a semantic algorithm to match inputs and outputs of Web service requests with inputs and outputs of Web service advertisements during the matchmaking process. It adds an additional mapping layer over UDDI and uses DAML-S as service description language to provide better service discovery than keyword based search.

The latest version (draft release 0.7) of DAML-S suggests using a WSDL file along with a DAML-S description to represent a service. Our approach involves annotating and extending WSDL constructs with DAML+OIL ontological concepts. Since *ServiceProfile* is used by DAML-S to describe and discover a Web service, all our extensions aim to provide the same functionality as *ServiceProfile* for Web service discovery. Some of the details in the *ServiceProfile* like service provider details are already supported by UDDI, so we have not added them to WSDL. We have used an approach similar to Paolucci et al [14] to store the semantic information about inputs, outputs and operations of a WSDL description in UDDI. As argued earlier, our approach has the advantage of an ontology-based approach that fits better with existing industry norms and standards, rather than requiring new infrastructure as needed by DAML-S. While the matching algorithm provided in Paolucci et al [11], uses only inputs and outputs to search for required Web services, our discovery algorithm first selects the services using ontological concepts representing functionality of operations, and then uses inputs and outputs to prune the search. In the example WSDL file given in figure 3, both the operations *buyTicket* and c*ancelTicket* have the same inputs and outputs, but they have different functionalities, therefore, searching just based on inputs and outputs would lead to incorrect results. Our algorithm also recommends using other details like preconditions and effects from the WSDL file to ensure that the operation matches exact requirements.

Ogbuji [17] discusses representing WSDL in RDF instead of XML. Our approach uses DAML+OIL ontologies in RDFS format to add semantics to Web service descriptions. The WSMF architecture [18] discusses using semantics at different levels of Web services stack. It proposes a conceptual framework that provides a model to describe Web services and their composition. Their approach is not specific to any of the

[4] Functionality is specified using ontological concepts that map to operations
[5] Preconditions and effects are not shown as they are optional
[6] Our implementation uses UDDI Version 1 API. Hence we have grouped operations with its inputs and outputs using the *keyName* parameter instead of *keyedReferenceGroup*

industry standards as they aim to use mediation to adapt to any standard.

A significant amount of recent research has focused on effective discovery of services, which is the key required capability of the Web services framework. The discovery mechanisms suggested to improve keyword based discovery range from categorization and domain independent characterization of services [19] to better techniques exploiting semantic representations of the services. Trastour et al. [20] analyses the problem of matchmaking and highlights the need for metadata for better results and suggests requirements for advanced matchmaking as high degree of flexibility and expressiveness; ability to express semi-structured data; support for type and subsumption; ability to express constraints over ranges of possible values as well as definite values of a specification. They have also observed that UDDI does not allow much expressiveness and flexibility. With our approach we use DAML+OIL ontologies to add these capabilities to UDDI data structure. Our approach involves storing semantically annotated Web service descriptions in UDDI, and using that semantic information for querying. Bernstein et al [21] suggests using process models to capture service behavior. The process models are created by indexing services to nodes in the process ontology. They also provide a process querying language for logic based querying of the process ontology to retrieve the most relevant services. Our approach maps operations in Web service descriptions to ontological concepts that represent functionalities and querying is based on templates.

6. Conclusions and Future Work

In this paper, we have presented our approach of adding semantics to Web services descriptions for improved Web service discovery. The current approach involves using DAML-S for adding semantics to Web services description. Since WSDL and UDDI are current industry standards, we believe that a pragmatic solution for adding semantics to Web services would be to add semantics to both of them, instead of creating a new language. While DAML-S is a highly expressive language, in which the features are meant not only for discovery but also for composition, execution and monitoring. In this paper, we have concentrated on adding enough semantics to WSDL using DAML+OIL ontologies, for it to have the same descriptive power as DAML-S for discovery. We have also discussed an algorithm for semantic discovery of Web services which uses functionality of the service as the main criterion for search. The main contributions of this work are

- Using the extensibility feature of WSDL to add semantics to service descriptions
- Using UDDI data structures to represent grouping of operations with their inputs and outputs

As part of the ongoing METEOR-S project, we are currently working on enhancing WSDL to make them better suited for service selection in e-commerce. Some of the features we intend to add are functional and behavioral attributes like QoS and constraints. We are also working on developing richer ontologies to depict functionality of operations [20], preconditions and effects. To fully utilize the potential of DAML+OIL ontologies, which we use for adding semantics to WSDL and UDDI, we are working on developing a more powerful logic based querying mechanism.

7. References

[1] Curbera, F., Nagy, W. and Weerawarana, S. “Web Services: Why and How.” Workshop on Object-Oriented Web Services – OOPSLA 2001, Tampa, Florida, USA, October 2001.

[2] Chinnici, R., Gudgin, M., Moreau, J. and Weerawarana, S. “Web Services Description Language (WSDL) Version 1.2 ", W3C Working Draft 24 January 2003, Available at http://www.w3.org/TR/2003/WD-wsdl12-20030124/

[3] Universal Description, Discovery and Integration: UDDI Technical White paper. 2000. http://www.uddi.org/pubs/Iru_UDDI_Technical_White_Paper.pdf

[4] Sheth, A. and Meersman, R. “Amicalola Report: Database and Information Systems Research Challenges and Opportunities in Semantic Web and Enterprises,” ACM SIGMOD Record, Vol. 31, No. 4, December 2002, pp. 98-106.

[5] Gruber, T.R. “A Translation Approach to Portable Ontology Specifications.” Knowledge Acquisition, 5(2), 199-220, 1993.

[6] Gruber, T.R. 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

[7] Kashyap, V. and Sheth, A. Semantics-based Information Brokering. In Proceedings of the Third International Conference on Information and Knowledge Management (CIKM), November 1994.

[8] Wache, H., V ogele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hubner, S. Ontology-based integration of information - a survey of existing approaches. In Stuckenschmidt, H., ed., IJCAI-01 Workshop: Ontologies and Information Sharing, 2001, 108--117.

[9] 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. The DAML Services Coalition, "DAML-S: Web Service Description for the Semantic Web", The First International Semantic Web Conference (ISWC), Sardinia (Italy), June, 2002.

[10] DAML-S 0.7 Draft Release, 2002.

[11] Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K. "Semantic Matching of Web Services Capabilities." Forthcoming in Proceedings of the 1st International Semantic Web Conference, 2002.

[12] Duftler, M.J., Mukhi, N.K., Slominski, A. and Weerawarana, S. Web Services Invocation Framework, 2001.

[13] Patil, A., Oundhakar, S. and Sheth, A. Semantic Annotation of Web Services, Technical Report, LSDIS Lab, Department of Computer Science, University of Georgia, March 2003.

[14] Paolucci, M., Kawamura, T., Payne, T.R. and Sycara, K. "Importing the Semantic Web in UDDI." To Appear In Web Services, E-Business and Semantic Web Workshop, 2002.

[15] Cardoso, J. and Sheth A. (2002). "Semantic e-Workflow Composition." Journal of Intelligent Information Systems (JIIS). (under revision).

[16] 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 (submitted). http://www.cs.uga.edu/~verma/METEOR-S-WSDI-submit.doc

[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] Bussler, C., Fensel, D. and Maedche, A. A Conceptual Architecture for Semantic Web Enabled Web Services SIGMOD Record, Special Issue Semantic Web and Databases 2002/12/01

[19] Dumas, M., O'Sullivan, J., Heravizadeh, M., Edmond, D. and Hofstede, A. Towards a Semantic Framework for Service description. In Proc. of the 9th Int. Conf. on Database Semantics, Hong-Kong, April 2001. Kluwer Academic Publishers.

[20] Trastour, D., Bartolini, C. and Gonzalez-Castillo, J. 2001. A Semantic Web Approach to Service Description for Matchmaking of Services, Proceedings of the International Semantic Web Working Symposium (SWWS)

[21] Bernstein, A. and Klein, M. 2002. "Discovering Services: Towards High Precision Service Retrieval" in Proceedings of the CaiSE workshop on Web Services, e-Business, and the Semantic Web: Foundations, Models, Architecture, Engineering and Applications. Toronto, Canada

```
<businessService businessKey="uddi:LSDIS_Travel.example" serviceKey="…">
   … <categoryBag>
        <keyedReferenceGroup tModelKey= "uddi:ubr.uddi.org:categorizationGroup:MAPPINGGROUP">
             <keyedReference tModelKey="uddi:ubr.uddi.org:categorization:OPERATION_CONCEPTS"
                 keyName="buyTicket" keyValue="TicketBooking"/>
             <keyedReference tModelKey="uddi:ubr.uddi.org:categorization:INPUT_CONCEPTS"
                 keyName="Input" keyValue="TicketInformation"/>
             <keyedReference tModelKey="uddi:ubr.uddi.org:categorization:OUTPUT_CONCEPTS"
                 keyName="Output" keyValue="ConfirmationMessage"/>
        </keyedReferenceGroup>
        <keyedReferenceGroup tModelKey="uddi:ubr.uddi.org:categorizationGroup:MAPPINGGROUP">
             …… </keyedReferenceGroup>
   … </categoryBag>
</businessService>
```

Figure 1: Representation of Operations, Inputs and Outputs in UDDI

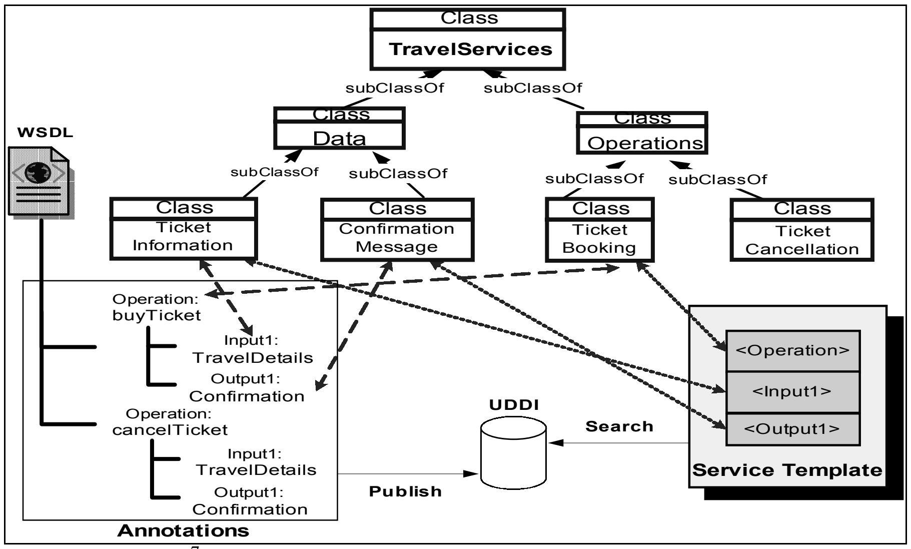

Figure 2[7]: Semantic Annotation, Publication and Discovery

[7] For simplicity of depicting, TravelService Ontology is shown in figure 2 with separate classes called data and operations, meaning TicketInformation or ConfirmationMessage are of type data, TicketBooking or TicketCancellation are of type Operations.

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
targetNamespace="http://decatur.cs.uga.edu:8080/axis/services/LSDISTravelWebService/axis/services/LSDISTravelWebService"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  .....
  xmlns:LSDISOnt="http://lsdis.cs.uga.edu/proj/meteor/METEORS/TravelServiceOntology.daml"
  xmlns:LSDISExt="http://lsdis.cs.uga.edu/proj/meteor/METEORS/WSDLExtension"
  .....
 <schema targetNamespace="http://LSDIS" xmlns="http://www.w3.org/2001/XMLSchema">
   <import namespace="http://schemas.xmlsoap.org/soap/encoding/"/>
   <complexType name="TravelDetails">
    <sequence>
    <element name="TravellerName" type="string"/>
    <element name="TravelType" type="string"/>
     <element name="FlightCode" type="string"/>
     <element name="CreditCardNo" type="int"/>
     <element name="OriginAirportCode" type="string"/>
     <element name="DestinationAirportCode" type="string"/>
     <element name="TravelDate" type="date"/>
    </sequence>
  </complexType>
  .....
  </schema>
</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-ReadyForPickUp"/>
  </wsdl:operation>
  <wsdl:operation name="cancelTicket" parameterOrder="in0" LSDISExt:operation-concept="LSDISOnt:TicketCancellation">
     <wsdl:input message="intf:OperationRequest" name="cancelTicketRequest"/>
     <wsdl:output message="intf:OperationResponse" name="cancelTicketResponse"/>
     <LSDISExt:precondition name="CreditCardValidity" LSDISExt:precondition-concept="LSDISOnt:ValidCreditCard"/>
     <LSDISExt:precondition name="TicketBookedBefore" LSDISExt:precondition-concept="LSDISOnt:TicketExists"/>
     <LSDISExt:effect name="TicketCancelled" LSDISExt:effect-concept="LSDISOnt:CardCredited"/>
  </wsdl:operation>
  </wsdl:portType>
  .....
  <wsdl:service name="LSDISTravelService">
    <wsdl:port binding="intf:LSDISTravelWebServiceSoapBinding" name="LSDISTravelWebService">
      <wsdlsoap:address location="http://decatur.cs.uga.edu:8080/axis/services/LSDISTravelWebService"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

Figure 3: WSDL File Extended[8] with Semantic Constructs

[8] a. The extended elements and attributes have been underlined and are optional.
b. Ontologies are common for data (inputs and outputs) and are not typical for operations, preconditions and effects. Hence to enable use of standard vocabularies and business terminologies, *onto-concept* attribute is not used with them.