Exploiting Semantic of Web Services through ebXML Registries

Asuman DOGAC
Software Research and Development Center, Middle East Technical University, Inonu Bulvari, METU(ODTU) Campus, 06531, Ankara, Turkey Tel:+90 312 2105598
asuman@srdc.metu.edu.tr, [1]

Abstract: The aim of this tutorial is to present how Web service semantics can be exploited through ebXML registries. The tutorial starts with basic concepts including XML, Web services, Web service standards and builds on top of these concepts to address how to exploit service semantics in ebXML registries through simple but comprehensive examples. The aim is to make these topics easily digestible to the audience so that the companies can judge for themselves the possible benefits of these technologies.

1. Introduction

Since Extensible Markup Language (XML) [16] is the common base of the technologies described in this tutorial, first a brief introduction to XML is presented. Over the recent years XML has become the "universal" standard for representing data. Starting out as a standard data exchange format for the Web, it has quickly become the fundamental instrument in the development of Web-based systems and standards.

1.1 XML in Brief

EDI	XML
BGM+220+**1234ABCD**+9' DTM+137:**20030601**:102' LIN+1' PIA+5+**9344**:EN+1078341ITEM:VP QTY+21:**16**:EA' PRI+AAA:**95**' LIN+2' …	<PurchaseOrderRequest> <PONumber>**1234ABCD** </PONumber> <PurchaseOrderDate>**20030601**</PurchaseOrderDate> <FirstLineItem> <ItemEAN_Identification no=**9344** /> <QuantityOrdered> **16** </QuantityOrdered> <UnitPrice> **95** </UnitPrice></FirstLineItem> <SecondLineItem> …

Figure 1. Example Purchase Order Documents in EDI and XML

XML makes use of **tags** to give meaning and structure to data. Consider the XML example given in Figure 1. We give the structure of "PurchaseOrderRequest" document by defining its sub elements such as "PONumber" and "PurchaseOrderDate". With the tag "PONumber", we give a meaning to the string "1234ABCD", that this is a purchase order number. Similarly "PurchaseOrderDate" tag gives a meaning to the string "20030601" that it is a purchase order date. In Figure 1, we also provide the corresponding EDI message [7]. From this example, it is clear that:

- An XML document is a text document in contrast to a binary file which requires specialized software to process it.

[1] This work is supported in part by the Scientific and Technical Research Council of Turkey, Project No: EEEAG 102E035

- An XML document is human readable. That is, although it is processed through software, when necessary, a human can read it to make sense out of it.
- There are several public domain, open source tools to process XML documents such as editors and parsers.
- As long as the communicating parties agree on the tags, XML documents are machine processable. That is, if two parties agree on the structure and tags in the XML documents, they can write programs that will accept these XML documents as input and process them automatically.

1.2 Why XML is useful?

In order to explain why XML is useful, we compare some of the basic features of XML technology with that of EDI as shown in Figure 2.

XML	EDI
XML is an open human-readable, text format.	EDI documents are typically in a compressed, machine-only readable form (Please refer to Figure 1).
XML documents are typically sent via the Internet - i.e. a relatively low-cost public network.	EDI documents are typically sent via private and relatively expensive value-added networks (VANs).
XML has low ongoing flat-rate costs using existing Internet connections and relatively low-cost Web Servers.	EDI can involve high on-going transaction based costs keeping up the connection to the EDI network and keeping the servers up and running.
XML is being developed in a world of shared software development populated by many low-cost tools and open source projects.	EDI was traditionally built from the ground up in semi-isolation without being able to share resources with other programs.

Figure 2. A Comparison of XML technology with EDI

1.3 What is a Web service?

A Web service is a business function made available via Internet through XML artifacts by a service provider and accessible by clients that could be human users or software applications. Any application/component can be exposed as a Web Service. For example, one can have a Web service to accept purchase orders automatically.

The revolutionary aspect of Web services is that they provide interoperability at the interface level. This allows for clean integration across departments, organizations, and companies. The client who invokes the service and platform hosting the Web service can be different; they can be using different programming languages and operating systems. Note that one Web Service can make use of other Web services to perform a complex function.

Interactions among Web services involve three types of participants: service provider, service registry and service consumer as shown in Figure 3. Web services are stored in service registries. The universal standard for the technical specification of Web services is WSDL (Web Services Description Language) [15] and the standard for invoking Web services is SOAP (Simple Object Access Protocol) [11] which provides an XML based messaging and remote procedure call (RPC) mechanism.

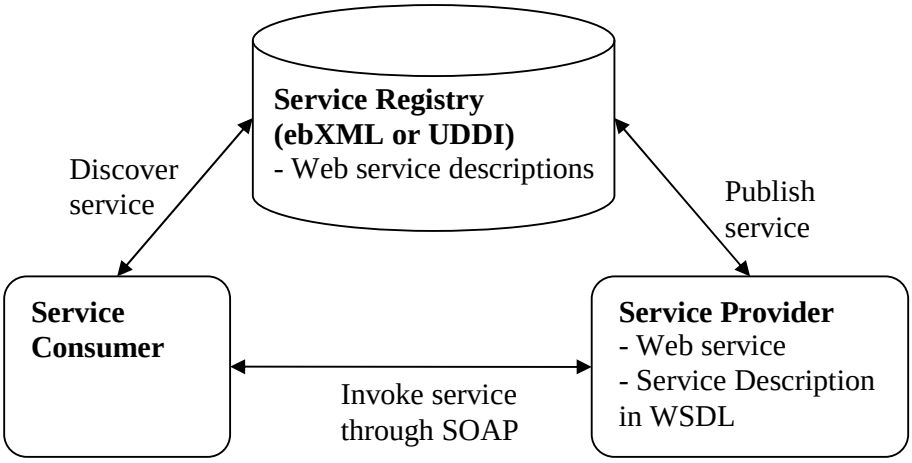

Figure 3. Web Service Model

There are two well known service registries: Electronic Business XML (ebXML) [6] Registries and the The Universal Description, Discovery, Integration framework (UDDI) [12] Registries. In this tutorial, we concentrate on ebXML.

Electronic Business XML is an initiative from OASIS and United Nations Centre for Trade Facilitation and Electronic Business [13]. ebXML aims to provide the exchange of electronic business data in Business-to-Business and Business-to-Customer environments. The ebXML specifications provide a framework in which EDI's substantial investments in Business Processes can be preserved in an architecture that exploits XML's technical capabilities. In other words, the initiative leverages from the success of EDI in large businesses, and intends reaching small and medium enterprises. ebXML builds on the lessons learned from EDI, particularly the need to identify trading partners and messages, and account for all message traffic. ebXML also identifies common data objects, called core components, that allow companies to interchange standard EDI data with XML vocabularies compliant with the ebXML specifications. Note that a Web service in ebXML is represented with a “Service” class in ebXML Registry.

1.4 Why do we need the semantics of Web services?

Web Service Description Language (WSDL) specifies only the technical interface of the Web services. WSDL provides the signature of the operations of the service, that is, the name, parameters and the types of parameters of the service. In other words, WSDL does not have any mechanism to express the semantic of the services like what the service is about or its parameters.

On the other hand, Web services, like their real life counterparts, may have many properties such as:

- the methods of charging and payment,
- the channels by which the service is requested and provided,
- constraints on temporal and spatial availability,
- service quality, security, trust and rights attached to a service and many more.

As shown in Figure 4, to be able to describe these properties and later search for services according to their properties we need to describe the semantics of the service. This search needs to be done in a machine processable and interoperable manner. This in turn is possible only by describing the semantics of Web services through ontology languages. In other words, all the necessary properties of services can easily be defined through an ontology language and domain specific ontologies can be developed by standard bodies. It

is a good idea to ground domain specific ontologies in upper ontologies since in this way they are more consistent and it becomes easier to integrate them within distributed heterogeneous systems.

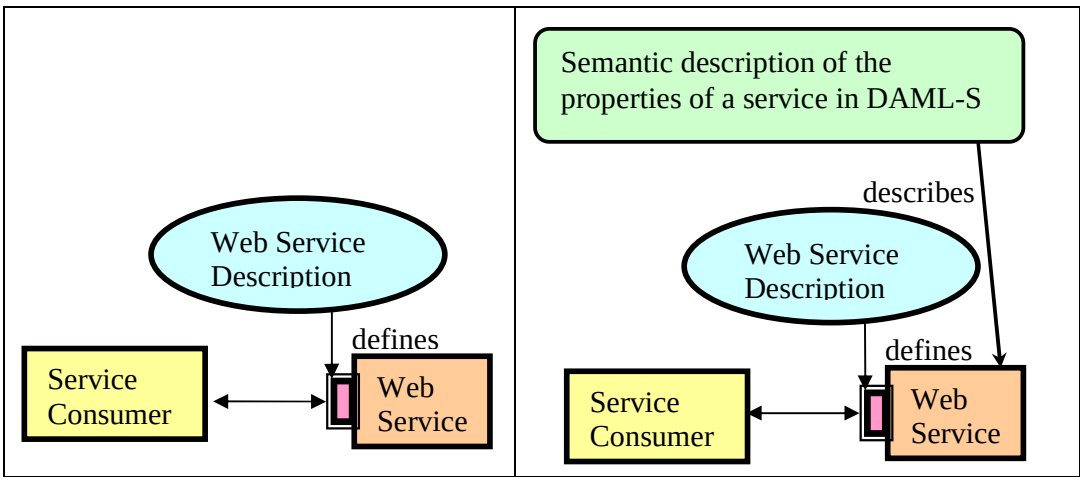

Figure 4. WSDL defines the interface of the service; semantic describes service itself

Currently, describing the semantic of Web in general [1], and semantic of Web services in particular are very active research areas. World Wide Web Consortium has started the initiative to develop Semantic Web and a semantic markup language for publishing and sharing ontologies, namely Web Ontology Language (OWL) [14], is being developed for this purpose. OWL is derived from DAML+OIL [2] by incorporating learnings from the design and application use of DAML+OIL. It builds upon the Resource Description Framework [9,10]. It should be noted that there are only minor differences between OWL and DAML+OIL.

There are a number of efforts for describing the semantics of Web services. Among these DAML-S [3] defines an upper ontology, that is, a generic "Service" class. In order to make use of DAML-S upper ontology, the lower levels of the ontology need to be defined. To provide interoperability, application domains must share such specifications. In fact, an ontology describes consensual knowledge, that is, it describes meaning which has been accepted by a group not by a single individual.

2. Describing the Semantics of Web Services, DAML-S

DAML-S defines an upper ontology for defining the semantics of Web services. It is based on DAML+OIL and aims to enable the automation of the following functionalities [3]:

- *Web service discovery*: Web service discovery involves the automatic location of Web services that provide a particular service and that adhere to requested constraints.
- *Web service invocation*: Web service invocation involves the automatic execution of an identified Web service by a computer program or a software agent.
- *Web service composition and interoperation*: This task involves the automatic selection, composition and interoperation of Web services to perform some tasks, given a high-level description of an objective.
- *Web service execution monitoring*: Individual services and, even more, compositions of services, will often require some time to execute completely. Users may want to know during this period what the status of their request is.

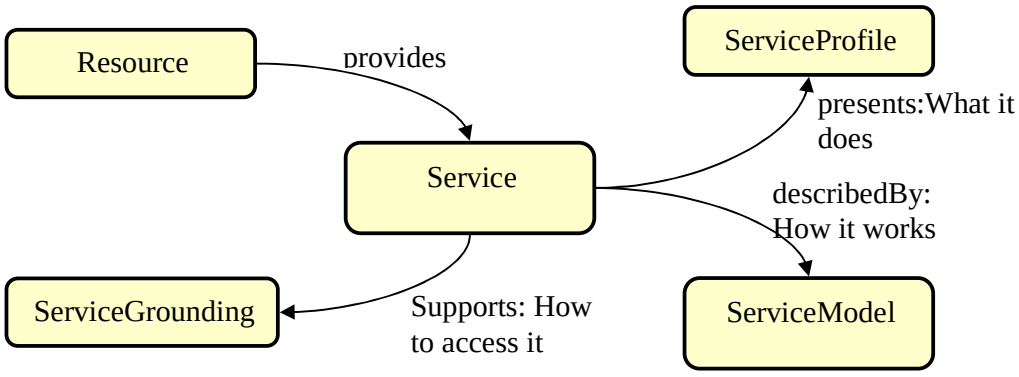

Figure 5. Describing the upper ontology of services

The top level class in DAML-S service taxonomy is the "Service" class as shown in Figure 5. Service class has the following three properties:

1. *presents*: The range of this property is *ServiceProfile* class. That is, the class Service *presents* a *ServiceProfile* to specify what the service provides for its users as well as what the service requires from its users.
2. *describedBy*: The range of this property is *ServiceModel* class. That is, the class Service is *describedBy* a *ServiceMode*l to specify how it works.
3. *supports*: The range of this property is *ServiceGrounding*. That is, the class Service *supports* a *ServiceGrounding* to specify how it is used.

2.1 How the do we define and use service semantics?

In relating the semantics with the services advertised in service registries, there are two key issues: the first one is where to store the generic semantics of the services (which could be a simple taxonomy or a more complex ontology). UDDI does not provide an internal mechanism to store generic service semantics. ebXML, on the other hand, through its classification hierarchy mechanism allows domain specific ontologies to be stored in the registries.

An ebXML registry is a mechanism where business documents and relevant metadata can be registered, and can be retrieved as a result of a query. A registry can be established by an industry group or standards organization.

Note that a service in ebXML is represented with a "Service" class in ebXML Registry. The technical specification files (i.e., WSDL descriptions of the service instance) are stored in ebXML registry together with "Service" class as extrinsic objects. The relationship between the description files and the "Service" class is established through the "ServiceBinding" Class of ebXML. A "Service" class may have a collection of "ServiceBinding" classes each of which represents technical information on how to access a specific interface offered by a "Service" instance. Also, a "ServiceBinding" instance has several "SpecificationLink"s each of which provides the linkage between a ServiceBinding and one of its specifications describing how to use the Service.

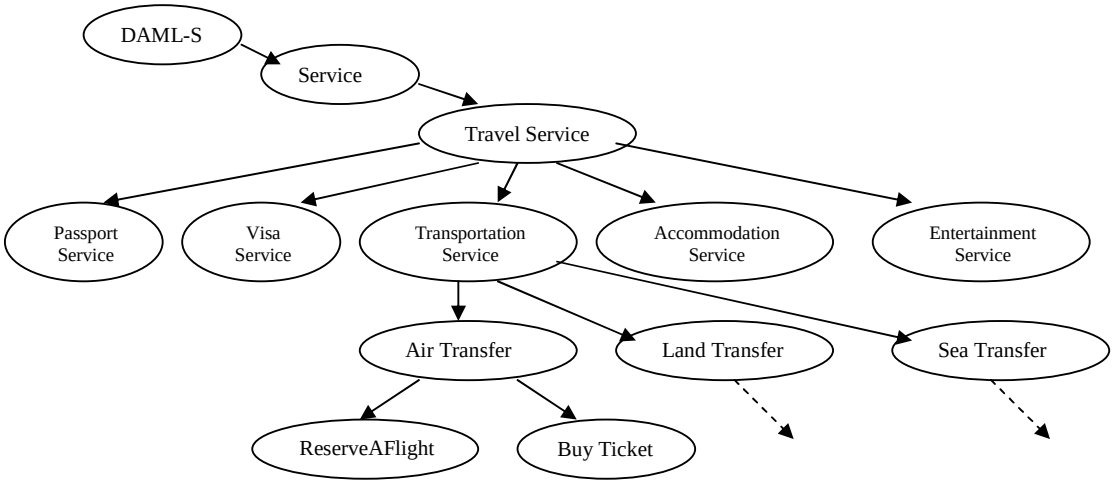

Figure 6 An example classification hierarchy for travel industry

An ebXML compliant registry allows metadata to be stored in the registry. This is achieved through a “classification" mechanism, called *ClassificationScheme* which helps to classify the objects in the registry. *ClassificationScheme* defines a hierarchy of *ClassificationNodes* [4,5].

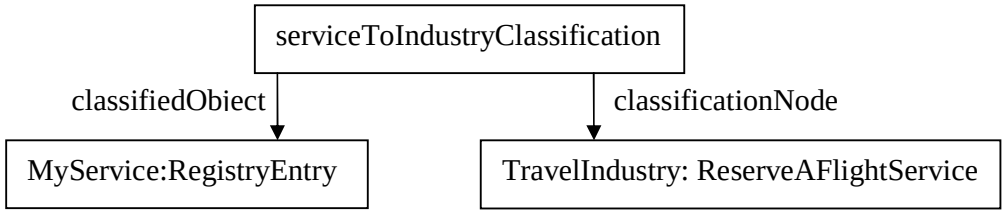

Figure 7 Relating service instances with the nodes in the classification hierarchy

Consider for instance the classification hierarchy example given in Figure 6 for travel industry. When such a hierarchy is stored in an ebXML registry, the registry objects can be related with the nodes in the hierarchy. In this way it is possible to give meaning to the services. In other words, by relating a service advertised with a node in classification hierarchy, we make the service an explicit member of this node and the service inherits the well-defined meaning associated with this node as well as the generic properties defined for this node. As an example, assume that there is a service instance in the ebXML registry, namely, “MyService”. When we associate “MyService” with “ReserveAFlightService” node as shown in Figure 7, its meaning becomes clear; that this service is a flight reservation service. Assuming that the “ReserveAFlightService” service has the generic properties such as “originatingFrom”, “destinationTo” and “paymentMethod” as shown in Figure 8, “MyService” also inherits these properties as shown in Figure 9.

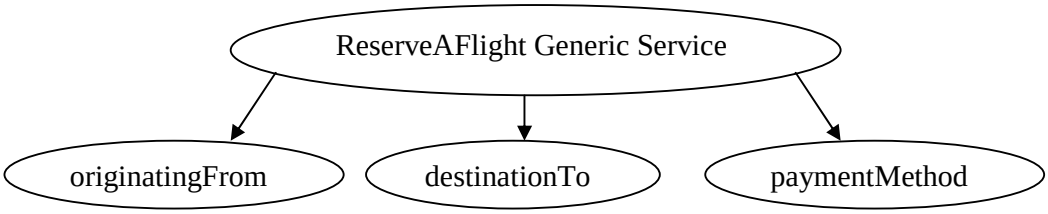

Figure 8. Properties of "ReserveAFlight" Generic Service

```
<!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns">
 <!ENTITY tron "http://www.srdc.metu.edu.tr/2003/TravelOntology.owl">
 <rdf:RDF
    xmlns:rdf =     "&rdf;#"
    xmlns:tron =    "&tron;#"
 <tron:ReserveAFlight rdf:ID="MyService">
  <tron:departureFrom>  Ankara </tron:departureFrom>
  <tron:destinationTo>  Bologna </tron:destinationTo>
  <tron:price>  500  </tron:price>
  <tron:paymentMethod>  Credit Card  </tron:paymentMethod>
  </tron:ReserveAFlight>
</rdf:RDF>
```

Figure 9. An Example Service Semantics

Providing precise meaning and properties of the services facilitate their discovery. For example, if we want to find all the flight reservation services in the registry, it is possible to query the services that are classified under the generic "ReserveAFlightService" node.

ebXML query facility can be used for this purpose which supports two query capabilities [5]: Filter Query and SQL Query. SQL Query support is optional whereas Filter Query is mandatory for a registry to be ebXML compliant. A client submits a Filter Query as part of an "AdhocQueryRequest".

2.2 What do we gain from service semantics?

When services are described according to a service ontology, it becomes possible to discover and compose services automatically. Consider for example the service definition given in Figure 9, where service properties are described in OWL by conforming to travel service ontology of Figure 6:

1. First, the service description is both human readable and machine processable. That is, one can write a program to find and process such services dynamically.
2. Secondly there are very many public tools such as APIs and reasoners to process such descriptions in DAML and RDF. OWL tools are expected to appear shortly. In other words, by conforming to a standard ontology language while describing semantics allows us to use publicly available tools.
3. Conforming to standards solves the interoperability problem; an organization complying with these standards can easily exchange machine processable information with other organizations.
4. The service name "MyService" does not convey information about what the service is about. However by making it an explicit member of the "ReserveAFlight" node in the ontology, we give it a concensually agreed meaning that it is a flight reservation service.

Hence, when a program is searching the service registry, it is able to identify this service as a flight reservation service.

5. Furthermore, we were able to describe the properties of this service which can be used in service discovery. For example, when an agent (human or software) searches a flight reservation service that accepts “credit card payment”, this service can easily be located since its “paymentMethod” contains this information.

2.3 What is the role of ebXML registry in supporting the semantics of Web services?

ebXML registry helps to support the semantic with the following mechanisms:

1. It is possible to define the properties of registry entries through “slots” which gives way to define the properties of Web services;
2. It is possible to store classification hierarchies in the services registry. Furthermore, ebXML allows relating the services advertised in the registry with the semantics defined in the classification hierarchy through explicitly declared classification objects. This makes it possible to query the registry to find out services according to their semantics.

Finally, the topics briefly mentioned over here due to space limitations will be elaborated during the tutorial presentation.

References

[1] Berners-Lee, T., Hendler, J., Lassila, O., “The Semantic Web", Scientific American, May 2001.

[2] DAML+OIL, http://www.w3.org/2001/10/daml+oil

[3] DAML Services Coalition (A. Ankolekar, M. Burstein, J. Hobbs, O. Lassila, D. Martin, S. McIlraith, S. Narayanan, M. Paolucci, T. Payne, K. Sycara, H. Zeng), “DAML-S: Semantic Markup for Web Services", in Proceedings of the International Semantic Web Working Symposium (SWWS), July 2001.

[4] ebRIM: ebXML Registry Information Model v2.1, June 2002, http://www.ebxml.org/ specs/ebRIM.pdf.

[5] ebRSS: ebXML Registry Services Specification v2.1, June 2002, http://www.ebxml. org /specs/ebiRS.pdf.

[6] ebXML, http://www.ebxml.org/

[7] Electronic Data Interchange, http://www.unece.org/cefact/index.htm.

[8] OASIS ebXML Registry Reference Implementation Project (ebxmlrr), http://ebxmlrr. sourceforge.net/

[9] RDF Schema: Resource Description Framework Schema Specification, W3C Proposed Recommendation, 1999, http://www.w3.org/TR/PR-rdf-schema.

[10] RDF Syntax: Resource Description Framework Model and Syntax Specification, W3C Recommendation, 1999, http://www.w3.org/TR/REC-rdf-syntax.

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

[12] UDDI: Universal Description, Discovery and Integration, http://www.uddi.org

[13] UN/CEFACT: United Nations Centre for Trade Facilitation and Electronic Business, http://www.unece.org/cefact/

[14] Web Ontology Language (OWL) Reference Version 1.0, http://www.daml.org/2002 /06/webont/owl-ref-proposed

[15] WSDL: Web Service Description Language, http://www.w3.org/TR/wsdl

[16] XML: Extensible Markup Language 1.0, W3C Recommendation, 1998, http://www. w3.org/TR/REC-xml-19980210.