A Semantic-Based Web Service Composition Facility for ebXML Registries

Asuman Dogac, Yildiray Kabak, Gokce B. Laleci

Software Research and Development Center, Middle East Technical University, Inonu Bulvari, Campus, 06531, Ankara, Turkey {asuman,yildiray,banu}@srdc.metu.edu.tr[1]

Abstract

Web services have become a main thrust of the IT industry and there has been considerable development in this area. There are two almost universally accepted standards: SOAP (Simple Object Access Protocol) for invoking services and WSDL (Web Services Description Language) for describing the technical specifications of the services. There are well-established service registries like UDDI (Universal Description, Discovery and Integration) by Microsoft and IBM, and ebXML (electronic business XML) by UN/CEFACT. The infrastructures for Web services are also readily available through application servers like IBM's WebSphere, Microsoft's .NET Framework or BEA's Weblogic. All of these application servers provide support for SOAP, WSDL and UDDI connectivity. However to be able to employ the full potential of Web services, their semantic information should be exploited. In this paper we describe a service composition tool which, by using the semantic information stored in ebXML registries, helps to automate the service discovery and composition.

Keywords

ebXML, ebXML classification hierarchies, Web service semantics, Web service composition, WSDL, SOAP

1 Introduction

The future of business-to-business interoperability seems to be in Web services. Through Web services Internet will become a global common platform where organizations and individuals communicate among each other to carry out various commercial activities and to provide value-added services. The dynamic enterprise and dynamic value chains become achievable and may be even mandatory for competitive advantage [Bussler, Fensel, Payne, Sycara, 2002].

The well accepted standards like Web Services Description Language (WSDL) [WSDL 2001], Simple Object Access Protocol (SOAP) [SOAP 2000] and service registries, such as UDDI [UDDI 2003] and ebXML [ebXML 2003] make it possible to dynamically invoke Web services. That is, when the service to be used is known, its WSDL description can be accessed from the registry through a program which uses the information in the WSDL description like the interface, binding and operations to dynamically access the service.

However to be able to exploit the Web services to their full potential, their semantic information should be available. An ebXML registry [ebRIM 2002] allows to define semantics basically through two mechanisms: first, it allows properties of registry objects to be defined through "slots" and, secondly, metadata can be stored in the registry through a "classification" mechanism. This information can then be used to discover the services by exploiting the ebXML query mechanisms.

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). Since UDDI does not provide an internal mechanism to store generic service semantics, we have chosen ebXML. ebXML, through its

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

classification hierarchy mechanism allows domain specific ontologies to be stored in the registries. Note that for UDDI registries, domain specific ontologies can be stored by the standard bodies who define them and the server, where the service is defined, can host the semantic description of the service instance.

The second key issue is how to relate the services advertised in the registry with the semantic defined through an ontology. The mechanism to relate semantics with services advertised in the UDDI registries are the tModel and the category bags of registry entries. tModels provide the ability to describe compliance with a taxonomy. The services can put the semantic information in their category bags through the tModels. In ebXML, on the other hand, classification objects explicitly link the services advertised with the nodes of a classification hierarchy.

In this paper, we describe a semantic-based service composition tool which automates service discovery and composition in ebXML registries. Through a graphical user interface provided, the tool allows the ebXML classification hierarchies to be depicted graphically. When a user clicks on a node in the classification hierarchy, the generic properties of the service are revealed to the user. The user can fill in the desired properties of services she is looking for through this GUI. The tool queries the ebXML registry automatically to find the services that satisfy user constraints. Then, through a graphical composition tool, the user is allowed to provide choreography of the selected services.

The paper is organized as follows: Section 2 briefly summarizes the ebXML specification. Section 3 presents the research approach. In Section 4, we describe the design and implementation of the system. Section 5 concludes the paper.

2 ebXML Specification

Electronic Business XML [ebXML 2003] is an initiative from OASIS and United Nations Centre for Trade Facilitation and Electronic Business [UN/CEFACT 2003]. 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. The vision of ebXML is to create a single set of internationally agreed upon technical specification that consists of common XML semantics and related document structures to facilitate global trade.

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.

The ebXML architecture provides the following functional components:

- Registry/Repository: A 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. A repository is a location (or a set of distributed locations) where a document pointed at by the registry resides and can be retrieved by conventional means (e.g., http or ftp).
- Trading Partner Information: The Collaboration Protocol Profile (CPP) provides both DTD and XML Schema definitions of an XML document that specifies the details of how an organization is able to conduct business electronically. It specifies such items as how to locate contact and other information about the organization, the types of network and file transport protocols it uses, network addresses, security

implementations, and how it does business (a reference to a Business Process Specification). The Collaboration Protocol Agreement (CPA) specifies the details of how two organizations have agreed to conduct business electronically. It is formed by combining the CPPs of the two organizations.

- Business Process Specification Schema: The Business Process Specification Schema provides the definition of an XML document (in the form of an XML DTD) that describes how an organization conducts its business. While the CPA/CPP deals with the technical aspects of how to conduct business electronically, the Business Process Specification Schema deals with the actual business process.
- Messaging Service: ebXML messaging service provides a standard way to exchange messages between organizations reliably and securely. It does not dictate any particular file transport mechanism, such as SMTP, HTTP, or FTP.
- Core Components: ebXML provides a core component architecture where a core component is a general building block that basically can be used to form business documents.

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.

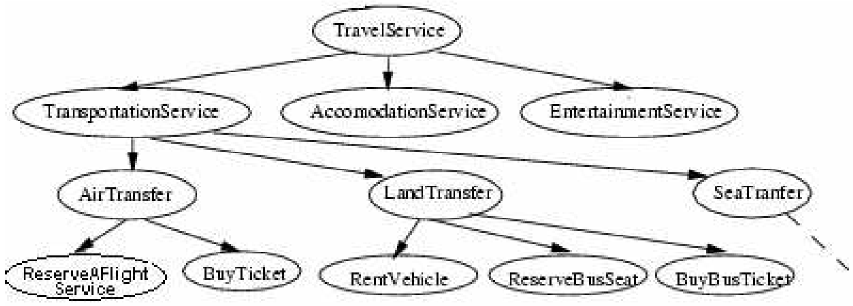


Figure 1 An example classification hierarchy for travel industry

3 The Research Approach

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* [ebRIM 2002, ebRSS 2002].

Consider for instance the classification hierarchy example given in Figure 1 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 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, 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”, “MyService” also inherits these properties.

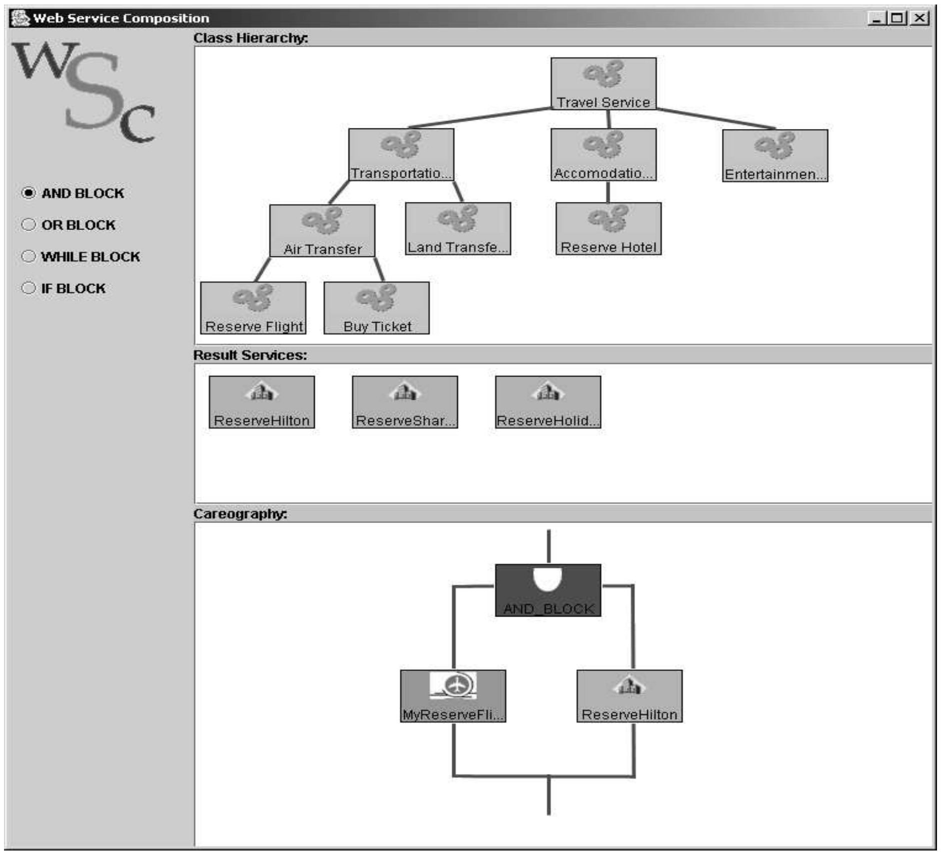
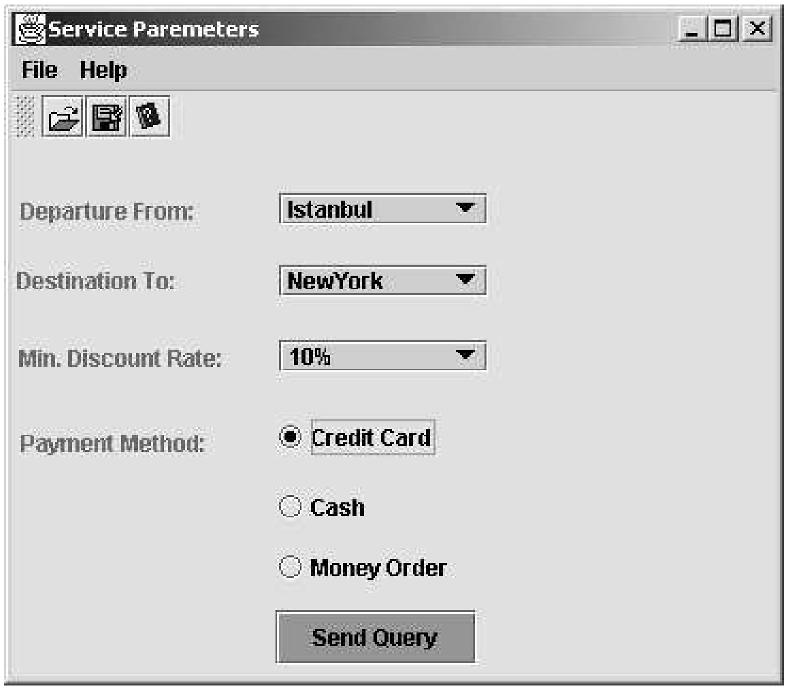


Figure 2 GUI tool for semantic-based Web service composition for ebXML registries



Departure From: Istanbul

Destination To: NewYork

Min. Discount Rate: 10%

Payment Method: [✓ Credit Card] Cash Money Order

Figure 3 GUI to obtain service property values from the user

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 [ebRSS 2002]: 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”. We have used Filter Query capability in the implementation.

What an “AdhocQueryRequest” must return is specified with the “ResponseOption”. The “ResponseOption" element has an attribute “returnType” whose values indicate the type of the response to be returned such as the identifiers of the registry objects, attributes of the registry objects, or leaf classes. For example, a RegistryObjectQuery with “ResponseOption” “LeafClass” will return all attributes of all instances that satisfy the query.

4 The Design and Implementation

We have developed a tool kit, whose GUI is presented in Figure 2. The top most part of the canvas depicts one of the ebXML classification hierarchies. When a user clicks on a classification node, the slots for this node are depicted to the user as shown in Figure 3 so that the user can provide the preferred values for the slots. Then the system finds the members of this node that satisfy the user given values and the resulting services are listed in the middle of the canvas. The user selects the services among the presented alternatives and forms a workflow by combining the services with the control flow blocks presented.

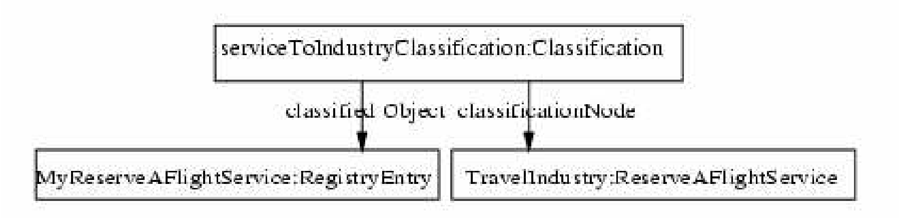

serviceToIndustryClassification:Classification
classified Object
classificationNode
MyReserveAFlightService:RegistryEntry
TravelIndustry:ReserveAFlightService

Figure 4 Relating service instances with the nodes in the classification hierarchy

In this section we present the details of this development effort. Each service is published to ebXML Registry via “SubmitObjectRequest”. Through “SubmitObjectRequest”, it is possible to relate the service instance with a generic node in the classification hierarchy. The WSDL files necessary to invoke the service is also indicated in this request.

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<SubmitObjectsRequest >
        <rim:LeafRegistryObjectList>
                <rim:ClassificationNode id = 'ReserveAFlightService' parent= 'CS' >
                        <Slot name = 'originatingFrom' slotType= 'StringList'>…</Slot>
                        <Slot name = 'destinationTo' slotType= 'StringList' > …</Slot>
                        <Slot name = 'paymentMethod' slotType= 'StringList' >…</Slot>
                </rim:ClassificationNode>
<Service id="MyReserveAFlightService">
        <Name> <LocalizedString lang="en_US" value = "Reserve Flight Service"/> </Name>
        <Slot name = 'originatingFrom'> <ValueList>
                <Value>Istanbul </Value>  </ValueList> </Slot>
        <Slot name = 'destinationTo'> <ValueList>
                <Value>New York</Value> </ValueList> </Slot>
                <Slot name = 'paymentMethod'> <ValueList>
                <Value>Credit Card</Value> </ValueList>    </Slot>
        <ServiceBinding  accessURI="http://www.sun.com/ebxmlrr/registry/nameSpaceIndexer">
                <SpecificationLink specificationObject="wsdl">
```

```
                    </SpecificationLink>
            </ServiceBinding> </Service>

<ExtrinsicObject id="wsdl" mimeType="text/xml">      </ExtrinsicObject>
<Classification classificationNode="ReserveAFlightService" classifiedObject="MyReserveAFlightService" />
            </rim:LeafRegistryObjectList>
</SubmitObjectsRequest>
```

Figure 5 “SubmitObjectRequest” which declares the semantic of “MyReserveAFlightService” and relates it with the “ReserveAFlightService”

As an example, assume that there is a service called “MyReserveAFlightService” stored in the ebXML registry and a *ClassificationNode* exist for “ReserveAFlightService”. Figure 4 demonstrates how this service is associated with this classification node by using the classification object. In Figure 5 we provide an example “SubmitObjectRequest” which declares the semantic of “MyReserveAFlightService” and relates it with the “ReserveAFlightService” generic node. Note that a registry object can be classified according to any number of classification schemes.

```
<?xml version = "1.0" encoding = "UTF-8"?>
<AdhocQueryRequest xmlns = "urn:oasis:names:tc:ebxml-regrep:query:xsd:2.0"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "urn:oasis:names:tc:ebxml-regrep:query:xsd:2.0 query.xsd">
  <ResponseOption returnType = "LeafClass" returnComposedObjects = "true"/>
  <FilterQuery>
    <RegistryObjectQuery>
      <NameBranch>
            <LocalizedStringFilter>
              <Clause>
                <SimpleClause leftArgument = "value">
                  <StringClause stringPredicate = "Equal">
                        ReserveAFlightService</StringClause>
                </SimpleClause>
              </Clause> </LocalizedStringFilter>
      </NameBranch>  </RegistryObjectQuery>  </FilterQuery> </AdhocQueryRequest>
```

Figure 6 An AdhocQueryRequest to return “ReserveAFlightService” as a composite object including the slots

Once the services are classified with classification nodes in this manner, it is possible to obtain all services that satisfy user given criteria by using ebXML’s query facilities. We have developed three query templates for this purpose:

- The first query template retrieves all the properties of a generic node. Given a node in the classification hierarchy, such as “ReserveAFlightService” node, this query, shown in Figure 6, returns the properties of the service class. Note that when the “ResponseOption” is “LeafClass” and “returnComposedObjects” is true, the properties (i.e., the slots) of the object are returned. The tool developed uses these properties to generate a dynamic GUI so that the user can provide her preferred values. An example GUI is provided for the “ReserveAFlightService” is given in Figure 3. Through this GUI, the user fills in her preferred values for the service parameters.
- The second query template is used for locating service instances of a given generic class node in the class hierarchy. Assuming that the user requested all the flight

reservation services originating from Istanbul, the query given in Figure 7 retrieves "ReserveAFlightService" instances "originatingFrom Istanbul".

- The third query template is a content retrieval query to obtain the WSDL files through the identifiers of the WSDL files in the SpecificationLinks.

```
<AdhocQueryRequest >
 <ResponseOption returnType = "LeafClass" returnComposedObjects = "true" />
  <FilterQuery>  <ServiceQuery>
   <ClassifiedByBranch>
      <ClassificationNodeQuery>  <NameBranch>  <LocalizedStringFilter>
          <Clause>
           <SimpleClause leftArgument = "value">
            <StringClause stringPredicate = "Equal">ReserveAFlightService </StringClause>
           </SimpleClause>
          </Clause>
        </LocalizedStringFilter>
       </NameBranch>
      </ClassificationNodeQuery>
    </ClassifiedByBranch>
          <SlotBranch>  <SlotFilter> <Clause>  <SimpleClause leftArgument = "name_">
                    <StringClause stringPredicate = "Equal">originatingFrom</StringClause>
                                        </SimpleClause> </Clause> </SlotFilter>
                    <SlotValueFilter> <Clause> <SimpleClause leftArgument = "value">
                    <StringClause stringPredicate = "Contains">Istanbul</StringClause>
                                        </SimpleClause> </Clause> </SlotValueFilter>
          </SlotBranch>
          …
   </ServiceQuery>   </FilterQuery> </AdhocQueryRequest>
```

Figure 7 An example ebXML query retrieving instances of a classification node

Finally, the retrieved services are depicted to the user and the user is allowed to form a workflow of selected services.

4.1 Implementation Status

A proof of concept implementation of the system is realized by using OASIS ebXML Registry Reference Implementation [ebXMLRR 2003]. As an application server to host Web services to be accessed through SOAP, Apache Tomcat 4.1 [Tomcat 2003] is used. The WSDL descriptions of the implemented services are generated through IBM Web services Toolkit 3.2 (WSTK) [WSTK 2000]. IBM Web Services Invocation Framework 1.0 (WSIF) [WSIF 2001] is used to invoke services.

5 Conclusions

Web Services will transform the web from a collection of information into a distributed device of computation. In order to employ their full potential, appropriate semantic mechanisms for web services need to be developed. In this paper we describe a semantic-based service composition tool for ebXML registries.

Service discovery and composition by using the semantics of services has been addressed in the literature where the weight of the work has been on using AI techniques to match the inputs and outputs of services requested and advertised. The main problem with these approaches is the performance, which is not yet at the levels to be adopted by the industry.

In contrast to such approaches, we take a data management approach for service discovery by exploiting the meta data and the query mechanism of the ebXML registry. With the tools we have provided, the registry is queried for explicit members of a classification node to obtain the generic parameters of the service and users are allowed to fill these in through graphical user interfaces generated dynamically. Then the registry is queried for matching services; the corresponding WSDL files are obtained and through a workflow mechanism presented, the users can compose the discovered services. The current workflow mechanism implemented is a simple one realizing basic workflow functionality. As a future work, we plan to use one of the recent Web services choreography languages, such as BPEL4WS [BPEL4WS 2002] or WSCI [WSCI 2002] in composing the services.

References

BPEL4WS: Business Process Execution Language for Web Services, http://www-106.ibm.com/developerworks/webservices/library/ws-bpel/

Bussler C., Fensel D., Payne T., Sycara K.: "ISWC Tutorial: Semantic Web Services", http://www.daml.ri.cmu.edu/tutorial/iswc-t3.html\#2b

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

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

ebXML: http://www.ebxml.org/

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

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

Tomcat: http://jakarta.apache.org/.

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

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

WSCI: Web Service Choreography Interface, http://wwws.sun.com/software/xml/developers/wsci/

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

WSIF: IBM Web Services Invocation Framework, http://www.alphaworks.ibm.com/tech/wsif

WSTK: IBM Web Services Toolkit (WSTK), http://alphaworks.ibm.com/tech/webservicestoolkit