Web Services Composition for Business Process Automation

YUEFENG FANG, KUN GAO, XIAOYONG WANG, JIFANG LI
Computer Science and Information Technology College
Zhejiang Wanli University
No. 8, South Qian Hu Road, 315100, Ningbo, Zhejiang
P. R. CHINA

Abstract: Composition of web services has received much interest to support business-to-business or enterprise application integration. Current standards to formalize the specification of web services, their flow composition and execution are introduced in this paper. Web services composition includes static and dynamic composition. We thoroughly introduced the static composition based on BPEL4WS through an example. Dynamic composition needs the support for automatic discovery ability. We put forward a method that can incorporate the semantic description into UDDI document based on UDDI's TModels. Thus we can take advantage of the UDDI popularity and support, while publishing semantically grounded descriptions of services that can be used to perform capability based search for services.

Key-Words: web services composition, business process automation, UDDI

1 Introduction

With the popularity of the World Wide Web, it has come the need for businesses to exploit the web not only for disseminating information but also for improving their interactions with their customers, distributors, suppliers and partners. The web service (WS) paradigm has emerged as an important mechanism for interoperation amongst separately developed distributed applications in the dynamic e-business environment[1].

Applications are to be assembled from a set of appropriate Web services and no longer be written manually. Seamless composition of Web services has enormous potential in streamlining business to business or enterprise application integration. In an ideal world, all enterprise e-business systems and applications would work together as a cohesive unit. Suppliers, business partners, departments, existing applications, and new e-business applications would interoperate and share data seamlessly. This would be a world of streamlined business processes and automated procedures, a world in which Internet and wireless technologies assure profitability and return on investment. Regardless of their geography or industry, all companies would be able to differentiate themselves from the competition, rapidly meet customer demands, and do business more efficiently with the best business partners[2].

Web Services can be combined to realize more complex processes or composite services. They are enabling to compose complex processes from services that are offered and implemented by different organizations. In this new model, the business world has developed a number of XML-based standards to formalize the specification of web services, their flow composition and execution. In section 2, the current standards for web services and their composition are introduced; the static composition of web services based on BPEL4WS is introduced in section 3; the semantic description can be incorporated into UDDI for dynamic web services discovery, which is introduced in section 4 and section 5 concludes this paper.

2 Current standards for Web Services

2.1 SOAP, WSDL and UDDI

There are three main components in the current web services architecture. Web languages such as Universal Description, Discovery, and Integration (UDDI) [3], Web Services Description Language (WSDL) [4] and Simple Object Access Protocol (SOAP) [5] define standards for service discovery, description and messaging protocols. The UDDI registry allows a business to publicly list a description of itself and the services it provides. Currently, IBM and Microsoft host UDDI nodes, HP and SAP are running beta nodes. Companies can register themselves with UDDI together with their web services. Other companies can then use UDDI to search for specific companies or web services.

2.2 Web Services Conversation Language

Hewlett-Packard developed the Web Services Conversation Language (WSCL), an XML-based specification layered on top of WSDL, for use in defining conversations between service providers and consumers. The Transition attribute defines the ordering between

interactions. The conversation proceeds from one interaction to another according to the legally defined transitions. Transition is unique to WSCL, whereas Interaction and Inbound/Outbound XML documents overlap with WSDL.

2.3 ebXML

The United Nations (UN/CEFACT) and OASIS sponsored the ebXML specifications for use in e-business frameworks. ebXML Business Process Specification Schema. ebXML BPSS is a proposed standard for specifying collaborations for use in exchanging business documents through a set of choreographed transactions[6].

2.4 Web Services Flow Language

IBM's Web Services Flow Language is an XML based specification for describing a public collaborative process and its compositions [7]. WSFL is layered on top of WSDL, which describes the service interfaces and their protocol bindings. It defines two types of web services compositions: a flow model specifies the execution sequence of a business process's functions and a global model combines flow models and provides a description of how the composed web services interact with each other.

2.5 XLANG

Microsoft's XLANG [8], an XML-based specification for describing executable business processes internal to a business, is layered on top of WSDL. The XLANG specification builds on the XML code for process description that Microsoft's Visiobased BizTalk Server Orchestration graphical modeling tool generates.

2.6 Business Process Modeling Language

The Business Process Management Initiative developed the XML-based BPML metalanguage for modeling executable private business processes. BPML is complementary to public collaborative process description languages, such as BPSS. It is based on the concept of transactional finite-state machines and has features that overlap XLANG.

2.7 Business Process Execution Language for Web Services

The Business Process Execution Language for Web Services is an XML-based language released by BEA, IBM, and Microsoft in July 2002, and it replaces the existing web service composition languages XLANG by Microsoft and Web Service Flow Language by IBM. It is used to describe executable business processes, which rely on the import and export of web services exclusively. The processes are specified on an abstract level disregarding any binding information of the service types to concrete service implementations. The specification of a business process within a BPEL4WS document mainly consists of three sections: the partners for the participants in the process interactions, the activities with the corresponding control-flow, and the containers for the required messages and their data-flow.

3 Static Web Services composition based on BPEL4WS

The fast and dynamic integration of business process is an essential requirement for organization to adapt their business practices to the dynamic nature of the Web. Business partners may need to form permanent (long term) or temporary (short term) relationships. In the former type of relationship, components are known in advance and alliances are statically defined. The latter form of partnership does not assume an a priori trading relationship among partners. An e-service would in this case need to dynamically discover partners to team up with to execute the required transactions. Thus, this type of dynamic integration (also called on-the-fly integration) requires support for automated partner discover and fast e-service integration.

In this section, we will discuss the static web services composition problems by describing a given example using BPEL4WS. Let us imagine an application scenario of booking travel packages in a travel agency. Upon receiving the customer order, the travel agent will create a trip request and derive the required hotel and flight reservations for it. It must cooperate with external specialized service providers that offer hotel and flight reservations. The process has to be integrated and all services must correctly interact with each other.

There are several interactions in this scenario. For example, "Customer to Create Itinerary" is abbreviated with CToCI, "Create Itinerary to Flight Service" is abbreviated with CIToFS, RIToFS stands for "Replan Itinerary to Flight Service". CIToHS stands for "Create Itinerary to Hotel Service". The following is a WSDL fragment for the travel agency service.

```
<definitions targetNamespace="http://..."
xmlns="http://schemas.xmlsoap.org/wsdl/">
<message name = "OrderEvent"></message>
<message name = "TripRquest"></message>
<message name = "FlightRequest"></message>
```

```
<message name = "HotelRequest"></message>
<message name = "BookingFailure"></message>
<portType name ="pt1">
<operation name ="CToCI">
<input message ="TripRequest"/>
</operation>
</portType>
<portType name ="pt2">
<operation name ="CIToHS">
<output message ="HotelRequest"/>
</operation>
</portType>
<portType name ="pt3">
<operation name ="CIToFS">
<output message ="FlightRequest"/>
</operation>
</portType>
...
<portType name ="pt9">
<operation name ="RIToFS">
<output message ="BookingFailure/>
</operation>
</portType>
</definitions>
```

The exact control and data flow that determines when an operation can execute is described in the following BPEL4WS language. It specifies the roles of each of the partners and the logical flow of the message exchanges.

```
<process name ="TripHandling">
<partners>
<partner name ="Customer"
myRole ="TripHandlingAgent"
serviceLinkType ="ExternalServiceLink"
partnerRole ="CustomerAgent"/>
<partner name ="FlightService"
myRole ="TripHandlingAgent"
serviceLinkType ="InternalServiceLink"
partnerRole ="FlightServiceAgent"/>
<partner name ="HotelService"
myRole ="tripHandlingAgent"
serviceLinkType ="InternalServiceLink"
partnerRole ="HotelServiceAgent"/>
</partners>
<containers> ... <containers>
...
</process>
```

The most difficult is to specify the logic of the message flow. BPEL4WS provides programming-language like constructs (sequence, switch, while, pick). The process starts when it receives a trip request from the customer. After the request has been received, hotel and flight request messages can be sent in any order to the two partner services.

```
<sequence>
<receive partner="Customer"
portType ="pt1"
operation ="CToCI"
container ="OrderEvent">
</receive>
<flow>
<invoke partner ="HotelService"
portType ="pt2"
operation ="CIToHS"
inputContainer ="HotelRequest">
</invoke>
<invoke partner ="FlightService"
portType ="pt3"
operation ="CIToFS"
inputContainer ="FlightRequest">
</invoke>
</flow>
```

After the partner services have been invoked, the process waits for the services to send the results of their booking operations, which again can arrive in any order.

```
<flow>
<receive partner ="HotelService"
portType ="pt4"
operation ="HSToEVAL1"
container ="HotelRequest">
</receive>
<receive partner ="FlightService"
portType ="pt5"
operation ="FSToEVAL1"
container ="FlightRequest">
</receive>
</flow>
```

4 Incorporating UDDI for Web services automatic composition

Universal Description Discovery and Integration (UDDI) is an industrial initiative whose goal is to create an Internet wide registry of web services. UDDI allows businesses to register their contact points, and the web services that they provide. UDDI supports the registration of attributes of services via a construct called TModel. A TModel is a form of metadata that provides a reference system for information about services. It allows the specification of additional attributes of the entities described in the UDDI repository. For instance services can specify that they are based on the WSDL specification by referring to a publicly known WSDL TModel. In general TModels have two functions: the first is to tag the type of service advertised and whether some

specific conventions on the use of the UDDI registry have been applied. The second is to provide abstract keys to be associated with a service specific value. For example, a service may specify its category using the North American Industry Classification System (NAICS) published by the US Census.

UDDI allows a wide range of searches of the registry: services can be searched by name, by location, by business, by bindings or by TModels. It enjoys the wide support of many prominent software and hardware companied that invested heavily in web services. It is becoming the de-facto standards repository of web services. Nevertheless, UDDI has some shortcoming especially in the search mechanisms that prevent the exploitation of its full capacities. It allows only a keyword-based search of businesses, and does not support any inference based on the taxonomies referred to by the TModels.

DAML-S provides a way to solve this problem, by allowing a semantic description and matching of services within UDDI [11]. Therefore, if we could translate the DAML-S representations into UDDI representations we combine the best of two worlds: we take advantage of the UDDI popularity and support, while publishing semantically grounded descriptions of services that can be used to perform capability based search for services.

Some provenance information like the name and address of the service provider in DAML-S profiles can be mapped directly to UDDI records. DAML-S specific attributes such as inputs, outputs, and so on are instead represented using the TModel mechanism described above. Business Services records use these TModels to index the values they store from the DAML-S Profile they intend to represent. As an example consider the case of a stock quote reporting service that takes as input a ticker symbol and returns as output the current quote. The representation of the inputs and outputs of such a services is shown in the following fragment.

CategoryBag
KeyedReference
KeyName=Input
KeyValue=financialOntology: ticker
TmodelKey=”UUID of the DAML-S Input TModel”
KeyedReference
KeyName=Output
KeyValue=financialOntology: Quote
TmodelKey=”UUID of the DAML-S Output TModel”

One advantage of the mapping described here is that it is completely embedded in UDDI. Furthermore, all the search functionalities provided by UDDI can be used to retrieve information about services that are represented as DAML-S services.

5 Conclusions

Starting from the current standards for web services and their composition, we discussed the static composition based on BPEL4WS. The goal with web Services is to allow an automated discovery and composition of services. For that, means and languages for machine-readable descriptions of services are necessary. The composition of the flow based on BPEL4WS is still manually obtained. The dynamic composition of services requires the location of services based on their capabilities and the recognition of those services that can be matched together to create a composition. We introduced a method to incorporate semantic description into UDDI document based on UDDI’ TModels. Thus we take advantage of the UDDI popularity and support, while publishing semantically grounded descriptions of services that can be used to perform capability based search for services.

6 Acknowledgment

This research was supported by Ningbo Natural Science Foundation under Grant No. 2007A610045.

References:

[1] Gisolfi, D. Web Services Architect Part 1: An Introduction to Dynamic e-Business, IBM, April 2001.

[2] Building A Fully Integrated, Extended Enterprise, 2001, BEA White Paper

[3] UDDI. The UDDI technical white paper, 2000, http://www.uddi.org/.

[4] E. Christensen, F. Curbera, G. Meredith. Web Services Description Language (WSDL) 1.1, 2001. http://www.w3.org/TR/2001/NOTE-wsdl-20010315.

[5] W3C. SOAP 1.2Working draft, 2001. http://www.w3c.org/TR/2001/WD-soap12-part0-20011217/.

[6] Arindam Banerji, Claudio Bartolini, etc., Web Services Conversation Language (WSCL) 1.0, 2002, 3, Hewlett-Packard Company, http://www.w3.org/TR/wscl10/

[7] Assaf ARKIN, Business Process Modeling Language (BPML), 2001, 3, BPMI.org, http://xml.coverpages.org/WD-BPML-20010308.pdf

[8] DAML-S Coalition, DAML-S: Web Service Description for the Semantic Web, ISWC01, 2002