Electronic Commerce Research, 3: 25–42 (2003)

XML-Based Advanced UDDI Search Mechanism for B2B Integration

LIANG-JIE ZHANG *, TIAN CHAO, HENRY CHANG and JEN-YAO CHUNG
{zhanglj,tian,hychang,jychung}@us.ibm.com
IBM T.J. Watson Research Center, 1101 Kitchawan Road, Route 134, Yorktown Heights, NY 10598, USA

Abstract

Exploring appropriate business applications published as Web Services in UDDI registries is a critical issue. Search for such applications should be effective in terms of time and uniformed in terms of interfaces. In this paper, an XML-based advanced UDDI exploring engine, Business Explorer for Web Services (BE4WS), that provides developers with standard interfaces for efficiently searching business and service information in single or multiple UDDI registries is presented. The engine processes the proposed UDDI Search Markup Language (USML) that specifies a search request comprising multiple queries, UDDI sources, search criteria, and aggregation operator. The core component of BE4WS, Advanced UDDI Search Engine (AUSE), aggregates search results from different UDDI registries based on the USML request and built-in intelligent search facilities with the ability to cross-reference multiple categories. An aggregation example using BE4WS is demonstrated to enable easy B2B integration.

Keywords: Business Explorer for Web Services (BE4WS), UDDI Search Markup Language (USML), Advanced UDDI Search Engine (AUSE), B2B integration

0. Introduction

The emergence of Web Services represents the next evolution of e-business. Web services are Internet-based, self-contained, modular applications that perform a specific business task and conform to a standard interfaces described in Web Services Description Language (WSDL) [7]. The standard interfaces ensure each of these self-contained business services can be easily integrated with other services to create a complete business process. This interoperability allows businesses to dynamically publish, discover, and aggregate a wide range of Web services over the Internet and beat the competitor with innovative products, business processes, and value chains. Therefore, exploring such business applications published as Web services in the Universal Description, Discovery and Integration (UDDI) registry is a critical issue. Search for such applications should be effective in terms of time and uniformed in terms of interfaces.

Web Services can be published to either a public or a private UDDI registry. The design of UDDI allows simplified forms of discovering for services and enables trading partners to publish data about themselves and their advertised Web services to specific categories.

* Corresponding author.

The current UDDI search is focused on one query including single or multiple search criteria such as business name, identifier, discovery URL, category, as well as service name, category, tModel, etc. as specified in the UDDI v2 specification [5]. However, such simple search may not be adequate for business search in general, and there are several issues. First issue has to do with the size of the result set. With a projected population of thousands of entities in UDDI registry, it is unlikely that searching for businesses that satisfy a particular set of criteria will yield a manageably sized result set. So it is crucial to come up with an efficient search engine for narrowing down the search and pinpoint only those Web services specified by the criteria. The second issue is the accuracy. When a specific category along with UDDI registration data is registered, only people who know how to search for the exact categories as specified in [5] will be able to find results.

In addition, all search engines supported by available UDDI registries search only one registry, i.e., its own, such as Microsoft's UDDI [4]; same is true with UDDIs for HP and for IBM. Therefore, at the present time, to aggregate information from multiple UDDI registries and other searchable sources such as WSIL (Web Services Inspection Language) [Ballinger et al., 1] requires issuing a few sequential search requests, with each targeted for a UDDI registry or data source. Thus, there is a need for an advanced Web services exploration mechanism to extend the current search capability and provide result aggregation among multiple UDDIs in an efficient and performance sensitive manner.

In this paper, Business Explorer for Web Services (BE4WS) [Zhang et al., 9] is proposed to search businesses and services information from UDDI registries in an efficient and standard way. A UDDI Search Markup Language (USML) is defined for business applications to efficiently search UDDI registries. Based on the defined USML, a framework of Advanced UDDI Search Engine (AUSE) is proposed to conduct the searching and result aggregation process.

In the remainder of this paper, the UDDI Search Markup Language (USML) is introduced in Section 1. Then Advanced UDDI Search Engine (AUSE) is proposed to perform effective UDDI exploring process in an efficient manner in Section 2. The USML construction and search criteria are described in Section 3. In Section 4, some aggregation operators are defined and illustrated. Further, two types of APIs are defined for BE4WS to allow developers to easily explore UDDI registries using standard interfaces in Section 5. In Section 6, a sample application of BE4WS is used to create an intelligent shipping agent to search UDDI registry based on two search queries defined in a USML for B2B integration. Some related works are described in Section 7. The conclusions of this effective UDDI search mechanism are given at the end of this paper.

1. USML (UDDI Search Markup Language)

USML is an XML-based language proposed to uniform the search query format and dramatically reduce requesting times in a search in that an USML-based search request incorporates multiple search queries, UDDI sources and aggregation operators.

Table 1 shows a sample USML with two search queries to search two UDDI registries. The first query is defined to look up Private UDDI Registry 1 (wsbi10) for business with

Table 1. Sample USML to search multiple UDDI registries.

```
<?xml version="1.0"?>
<!DOCTYPE Search SYSTEM "UDDISearch.dtd">
<Search>
   <ProcessId>9999</ProcessId>
   <Query>
      <Source>Private UDDI Registry 1</Source>
      <SourceURL>http://wsbi10/services/uddi/inquiryAPI</SourceURL>
      <BusinessName >UPS </BusinessName >
      <FindBy>Business</FindBy>
   </Query>
   <Query>
      <Source>Private UDDI Registry 2</Source>
      <SourceURL>http://wsbi5/services/uddi/inquiryAPI</SourceURL>
      <BusinessName >KEP</BusinessName >
      <FindBy>Business</FindBy>
   </Query>
   <AggOperator>OR</AggOperator>
</Search>
```

BE4WS Aggregation Result-> Business List

Business:	Description:	Key:	Operator:
UPS	USA. UPS, or United Parcel Service Inc., is a 93-year-old company whose history spans the bicycle to the Internet. Today the largest express carrier and largest package carrier in the world, UPS continues to develop the frontiers of logistics.	339CFD70-E84D-11D5-B61C-970107B90C83	wsbi10/services/uddi
KEPEX Fulfillment Center	KEPEX	8EF91680-EDDA-11D5-ADF1-850C07A41C41	wsbi5:80/services/uddi

Figure 1. Sample aggregation result page using USML.

names starting with UPS and the second to look up Private UDDI Registry 2 (wsbi5) for business with names starting with KEP. An aggregation operator OR is used to aggregate the search results from these two different UDDI registries, and this normally would have to be achieved by issuing two separate requests via UDDI4J APIs, with one to wsbi10 and the other to wsbi5 and then combine the two results.

The aggregation result, an USML response, from BE4WS using this sample USML is presented in a JSP (Java Server Page) page shown in Figure 1.

The detailed schema and construction of a USML will be given in Section 3. It is noted that there are many productivity gains for e-business application developers by using USML because it can reduce the time and efforts of the developers and relieve them from the mundane tasks in two ways. First, with USML, being a high-level script language, it eliminates the need for a developer to master the use of low-level APIs such as provided by UDDI client package like UDDI4J [6]. Second, Advanced UDDI Search Engine further described in Section 2, takes cares of constructing multiple individual search requests to multiple UDDI registries and aggregates the results.

2. XML-based Advanced UDDI Search Engine (AUSE)

The basic idea of AUSE is to aggregate search results from different UDDI registries based on the proposed USML and its supporting intelligent search facilities such as Instant Notification Broker, UDDI Source Dispatching Broker, and Information Aggregation Broker, which have both priori knowledge of the meanings of specific categories and the ability to cross-reference across multiple categories. The sample UDDI registries include one public UDDI registry and multiple private UDDI registries. The aggregation broker will parse and re-organize the returned results from different UDDI registries based on aggregation operators or rule-based scripts. The final result is represented as an XML response to the search requester.

The architectural diagram is shown in Figure 2. The following mechanisms are used to enhance the UDDI search capability:

(1) Cascading Search Mechanism. It is used for refining the search results at different levels of granularity. Service requesters can use USML to define criteria to filter search results. Additional filtering mechanism can be applied to the search results that are returned from different UDDI registries, such as finding service providers within a certain price range or location.
(2) UDDI Search Markup Language (USML). USML is an XML-based language proposed to uniform the search query format and dramatically reduce search time in a request. An example USML and structure has been introduced in Section 1.

The sections that follow describe the AUSE architecture and several of its major components as illustrated in Figure 2. AUSE parses USML-based search requests and triggers search commands for different UDDI registries and other information sources and then aggregates results for the search service requesters.

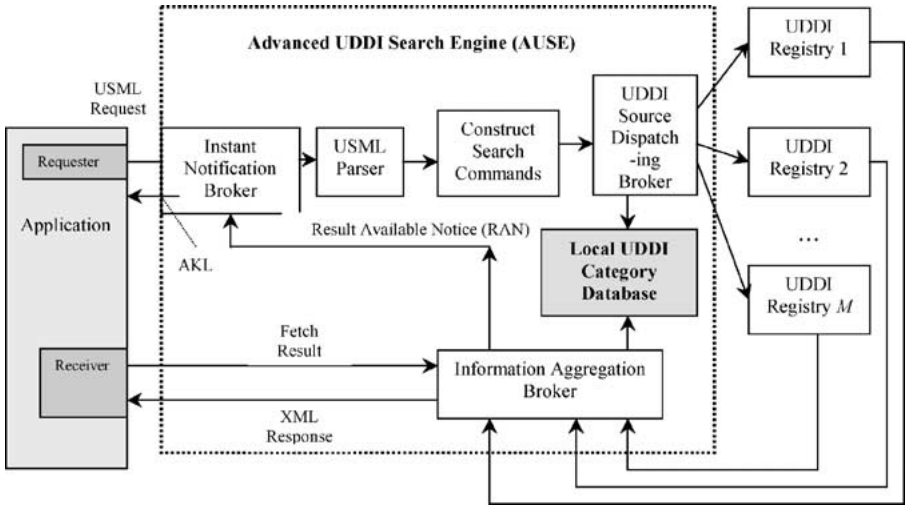

Figure 2. XML-based advanced UDDI Search System architecture.

Additionally, AUSE uses an Instant Notification Broker (INB) to communicate with the service requesters and Information Aggregation Broker. That is, when a USML-based query arrives, the AUSE will send an acknowledgment to the requester instantly. After the Information Aggregation Broker finishes the aggregation of search results from different UDDI registries, it will send out a Results Available Notice (RAN) to the INB. Then the INB will send search requester a notice so that the receiver in the application can fetch the results from AUSE as soon as possible. The second important component in AUSE is the UDDI Source Dispatching Broker, which is an intelligent broker that dynamically dispatches the multiple constructed UDDI search commands to the pre-selected UDDI registries based on the sources specified in a USML request. If no source specified, UDDI Source Dispatching Broker automatically dispatches the UDDI search commands to a best-known UDDI registry based on its knowledge base.

Moreover, UDDI search is a time-consuming process. To improve performance and enhance the categorization accuracy, another component, Local UDDI Category Database, is used to store and re-organize the UDDI categories based on its knowledge and self-updating mechanism. That is, the published category data from different UDDI registries can be used by the Local UDDI Category Database, which will be created above the UDDI technical layer. There are two ways that the Local UDDI Category Database can be updated: (1) in real-time when a search command is executed, or (2) on a periodical time-initiated basis where the AUSE updating mechanism automatically sends search commands to the available UDDI registries and organizes the returned results in a well-formatted way.

If a local category source is specified in the USML request, then the Source Dispatching Broker will route the search commands to the Local UDDI Category Database. Of course, one USML request can include multiple search commands specifying multiple data sources including Local UDDI Category Database, public UDDI Registry, and other private UDDI registries.

Network bandwidth is an extremely valuable resource for the networked solution providers, requesters as well as e-Marketplace. Therefore, from the system point of view, the proposed advanced UDDI search mechanism can reduce the network traffic from service requesters by issuing only one USML-based search request and receiving one XML-based response. In addition, it simplifies the developer's effort by avoiding having to master the lower level UDDI search programming skills. Additionally, a fast result can be returned from the AUSE if the Local UDDI Category Database is used.

In summary, AUSE can greatly increase the efficiency of e-business application development in supporting the business-level search for specific criteria such as finding partners with products in a certain price range or availability, or finding high quality partners with good reputations in an efficient way. In addition, the voluntary categorization data in UDDI may not be accurate at all time and therefore not sufficient to provide the precise results needed by e-business; in addition, there lacks the ability to cross-reference of categories. Thus, the Local UDDI Category Database is quite valuable in providing accurate search results for e-business search requesters.

3. USML construction

USML allows aggregation of multiple queries that searches the UDDI registries using multiple criteria. A search could be created to search businesses, services and serviceTypes, matching the different criteria as specified in USML. ServiceType is also called tModel in UDDI, and it is essentially a technical "fingerprint" unique to a particular specification. The USML response can be in one or all of the three basic data types, businessEntity, businessService, and ServiceType.

Document Type Definition (DTD) is a structural description of an XML document. It defines the elements an XML document can have, their attributes, their values and so on. A valid XML document must conform to the specified DTD. As shown in Table 2, *UDDISearch.dtd* associated with USML specifies the search criteria.

The description of each XML element is listed in Table 3, and several of them have the exact meaning and definitions as specified in the UDDI v2 specification [5], please refer to it for further details. AUSE also uses a configuration file to store large number of URLs associated with various UDDI registries. When the URL is not specified in USML, default URL associated with the Source element is taken from the configuration file where the Source UDDI names are mapped with their URLs. Thus, new registries can be easily added in this file at later stages without the need to modify the rest of the code to make use of the new registries.

The example in Table 4 shows how easy it is to use USML to construct multiple search criteria in one query to enable B2B integration. A user can selectively specify all or a subset of the criteria illustrated (not an exclusive list) to obtain the most precise services that meet the business criteria.

Table 2. USML DTD.

```
<!ELEMENT UDDISearch (ProcessId, Query+, AggOperator, RequestTypeName?, ReturnShortOrLong?)>
<!ELEMENT ProcessId (#PCDATA)>
<!ELEMENT Query (Source, SourceURL?, BusinessName?, Identifier?, Category?, ServiceName?,
                 ServiceTypeName?, DiscoveryURL?, FindQualifier?, TModelKey?, FindBy)>
<!ELEMENT Source (#PCDATA)>
<!ELEMENT SourceURL (#PCDATA)>
<!ELEMENT BusinessName (#PCDATA)>
<!ELEMENT Identifier (#PCDATA)>
<!ATTLIST Identifier type (D-U-N-S|ThomasRegister) #REQUIRED>
<!ELEMENT Category (#PCDATA)>
<!ATTLIST Category type (NAICS|UNSPSC|GEO|UDDITYPE|SIC|types) #REQUIRED>
<!ELEMENT ServiceName (#PCDATA)>
<!ELEMENT ServiceTypeName (#PCDATA)>
<!ELEMENT DiscoveryURL (#PCDATA)>
<!ELEMENT FindQualifier (#PCDATA)>
<!ELEMENT TModelKey (#PCDATA)>
<!ELEMENT FindBy (#PCDATA)>
<!ELEMENT AggOperator (#PCDATA)>
<!ELEMENT RequestTypeName (#PCDATA)>
<!ELEMENT ReturnShortOrLong (#PCDATA)>
```

Table 3. Description of each XML element in USML.

XML tag	Explanation
Query	It specifies the query conditions. It combines keyword search, search based on identifiers, and search based on categories.
Source	It is the name of UDDI source for the query. It can be "public UDDI" or "private UDDI" or other names.
SourceURL	It is the URL of the source so that BE4WS can access the registry and get related information.
BusinessName	It is the name of the business entity.
Identifier	It is the identifier name and its associated value. Two types of identifiers are supported: D-U-N-S, and ThomasRegister.
Category	It is the category name and its associated value. All category in the UDDI are supported: NAICS, UNSPSC, GEO, UDDITYPE, SIC and types.
ServiceName	It is the name of the service. It is used when the search is by service name.
ServiceTypeName	It is the name of the service type (i.e., tModel). It is used when the search is by the service type.
DiscoveryURL	It is the URL for discovery.
FindQualifier	It specifies the special behaviors used with searching, such as exactNameMatch or sortByNameDesc.
FindBy	It specifies the data type of the result value for one query. There are three basic data types in UDDI: Business, Service, and ServiceType (tModel).
AggOperator	It specifies the response of USML and the logic relationship among queries, i.e., as a union (denoted as "OR" operator) or an intersection (denoted as "AND" operator) of the multiple queries. Aggregation operators are further discussed in the following section.
RequestTypeName	It specifies the data type of the overall USML response. Either of the three basic UDDI data types specified in "FindBy" could be returned.
ReturnShortOrLong	It specifies the content of the USML response, either the short or detailed information of the UDDI data types.

Table 4. Sample USML for multiple search criteria.

```
<?xml version="1.0"?>
<!DOCTYPE Search SYSTEM "UDDISearch.dtd">
<Search>
  <ProcessId>9999</ProcessId>
  <Query>
    <Source>IBM_UDDIv2</Source>
    <SourceURL>http://www-3.ibm.com/services/uddi/v2beta/inquiryapi</SourceURL>
    <BusinessName>IBM Corporation</BusinessName>
    <Category type="NAICS">33461</Category>
    <Identifier type="D-U-N-S">00-136-8083</Identifier>
    <DiscoveryURL>http://www.ibm.com/services/uddi/uddiget?businessKey=D2033110-3AAF-11D5-80DC-
                  002035229C64</DiscoveryURL>
    <FindQualifier>exactNameMatch</FindQualifier>
    <FindBy>Business</FindBy>
  </Query>
  <AggOperator>OR</AggOperator>
  <ReturnShortOrLong>Long</ReturnShortOrLong>
</Search>
```

4. Aggregation operators

The "AggOperator" defined in USML can take different values such as union (OR), intersection (AND) operators, and script, a function which defines a script to perform a task. Currently, only "AND" and "OR" are supported. The results from different UDDI registries may be required to be aggregated depending on these operators. If the response contains redundant information, it can be filtered. For example, every Business is associated with a business key and every Service has a service key. Thus these operators help to combine the results of different keys and eliminate the repetitive information with the same key.

4.1. Union (OR) aggregation operator

With the union aggregation operator, it combines search results from multiple queries into one collective response and return that to the requester.

These are the several types of usages for the union aggregation operator: (1) aggregate data from multiple UDDI registries as the example shown in Table 1, which finds business from two different UDDIs; (2) aggregate data of different data types; (3) mix and match (1) and (2) to aggregate data from the same or different UDDIs, and the same or different data types. An example of aggregating data of various data types may be to search in a UDDI registry for any business starting with "IBM" plus services starting with "Web" specifying the Union aggregation operator. The USML response will include two data types, i.e., BusinessEntity and ServiceEntity. That is, with the union aggregation operator, only one USML request needs to be instead of two if traditional search methods are used to obtain the combined results of businesses and services, thus reducing the searching time and effort.

4.2. Intersection (AND) aggregation operator

On the other hand, the Intersection (AND) aggregation operator obtains results among all the queries that have data intersecting one another. For example, if we want to search for the service types starting with "Web" and these service types must be used by businesses whose names start with "White", we need to specify two queries: one for service type and one for business in one USML request. We use "AND" as the AggOperator tag and require the Service Type to be returned.

According to the UDDI specification, three core data types can be queried against: business, service, and service type (tModel). If the aggregation operator is "AND", the user is required to fill in the value for "RequestTypeName" that specifies one of the three core data types to be returned. For example, if "RequestTypeName" is Type, and three queries are specified in an XML document: one for business, one for service, and one for service type, only tModel information that meets all the requirements specified in these three queries is returned.

Thus, we can search by Businesses, Services and Service Types, and there are nine possible combinations of "AND" query. If we use "Type" to refer to "Service Type", and the

Table 5. Nine possible combinations of "AND" query.

Combination	Description
BusinessServiceType	AND of all three data types, returns Business
ServiceBusinessType	AND of all three data types, returns Service
TypeBusinessService	AND of all three data types, returns Type
BusinessService	AND of Business and Service, returns Business
ServiceBusiness	AND of Service and Business, returns Service
BusinessType	AND of Business and Type, returns Business
TypeBusiness	AND of Type and Business, returns Type
ServiceType	AND of Service and Type, returns Service
TypeService	AND of Type and Service, returns Type

first part of the following names as "RequstTypeName", the detailed description is illustrated in Table 5. The semantics of intersection or "AND" is as follows. For each "AND" query, the intersection of keys got from subqueries must not be empty. For example, if we use the value of RequestTypeName as "TypeBusiness", the returned service type must be used by at least one business specified in the query for "Business".

5. BE4WS Application Programming Interfaces (APIs)

BE4WS enables interaction with XML representations of UDDI exploring mechanism instead of direct work with UDDI for Java and other UDDI clients, which is the usual programming model. With BE4WS, the same programming model can be used regardless of how the UDDI for Java or other UDDI clients are implemented.

BE4WS consists of two types of APIs: a regular Java API and a Web Services Interface. With these APIs, the desired business entities, services, and service types (tModels) can easily be found from different UDDI registries using one single Java call or SOAP call. The only input parameter for these calls is the USML request string. The return result is also an XML-based USML response. These methods take the string containing the desired USML URL, USML file name or USML string as input and return a document of USMLResponse.

There are two steps to use these methods in BE4WS Java package: create a USML object with all the search criteria and write a Java application to include BE4WS package. Any e-business application developer can use this API to easily find the business entities and services from different UDDI registries using one single request.

BE4WS Web service can be published in the private or public UDDI Registry. Then any application can find this useful UDDI search service and invoke it through the WSDL Interfaces. You can easily use WebSphere Application Server (WAS) console to install BE4WS SOAP service as an enterprise application. The installed BE4WS can work as an advanced UDDI search portal that allows any SOAP clients to easily and effectively find business information and services information from multiple UDDI registries.

Figure 3 shows an installed BE4WS on WebSphere Application Server 4.0. It is an enterprise application. You can stop or start it using the WAS admin control. Further, you can assign the access roles and set a certain security level for BE4WS if required.

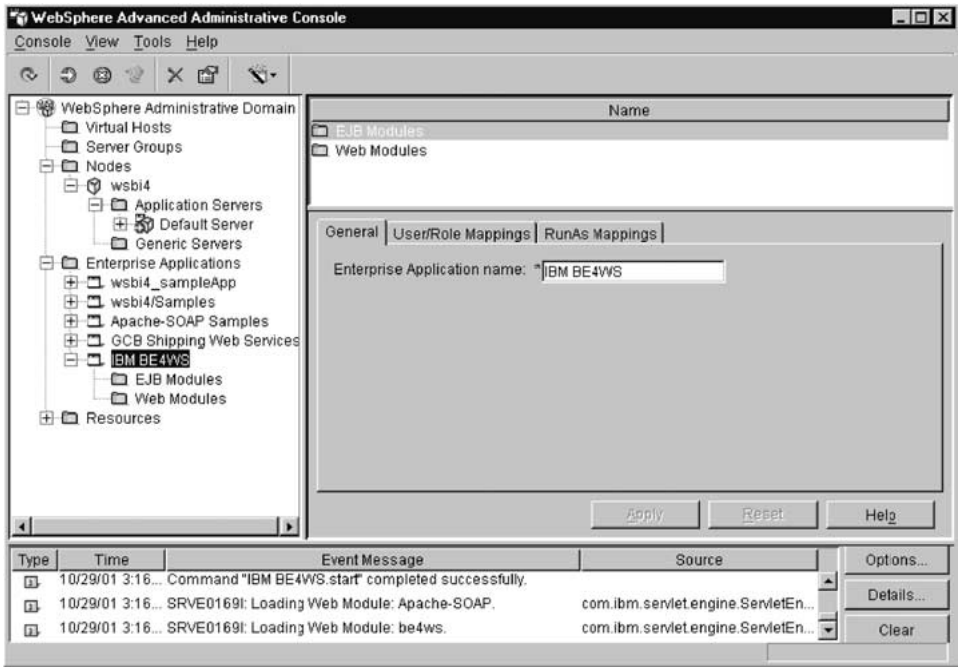

Figure 3. BE4WS installed on WAS 4.0.

After you install BE4WS on WebSphere Application Server, you can use the SOAP administration tool to list BE4WS service, start it and stop it. The deployed service information of BE4WS is shown in Figure 4. From Figure 4, we can find there are two methods in BE4WS SOAP service, i.e., searchByString, searchByURL. urn:uddisearch-service is the SOAP service ID, and com.ibm.UDDI.BusinessExplorer.UDDISearch is the provider class. An invocation client will use such information.

You can use regular SOAP RPC program to invoke BE4WS SOAP Service. The other way you can use is to take advantage of Web Services Invocation Framework (WSIF) [8]. It provides a standard API to invoke services; no matter how/where the service is provided as long it is described in WSDL. The architecture allows new bindings to be added at runtime. WSIF enables the user to move away from the usual Web Services programming model of working directly with the SOAP APIs, and towards a model where the user interacts with representations of the services. This allows the user to work with the same programming model regardless of how the service is implemented and accessed. More information about WSIF can be found from IBM alphaWorks [8]. The architecture of WSIF also allows stubless invocation of Web Services. That is, no stub is generated, and the services can be dynamically invoked.

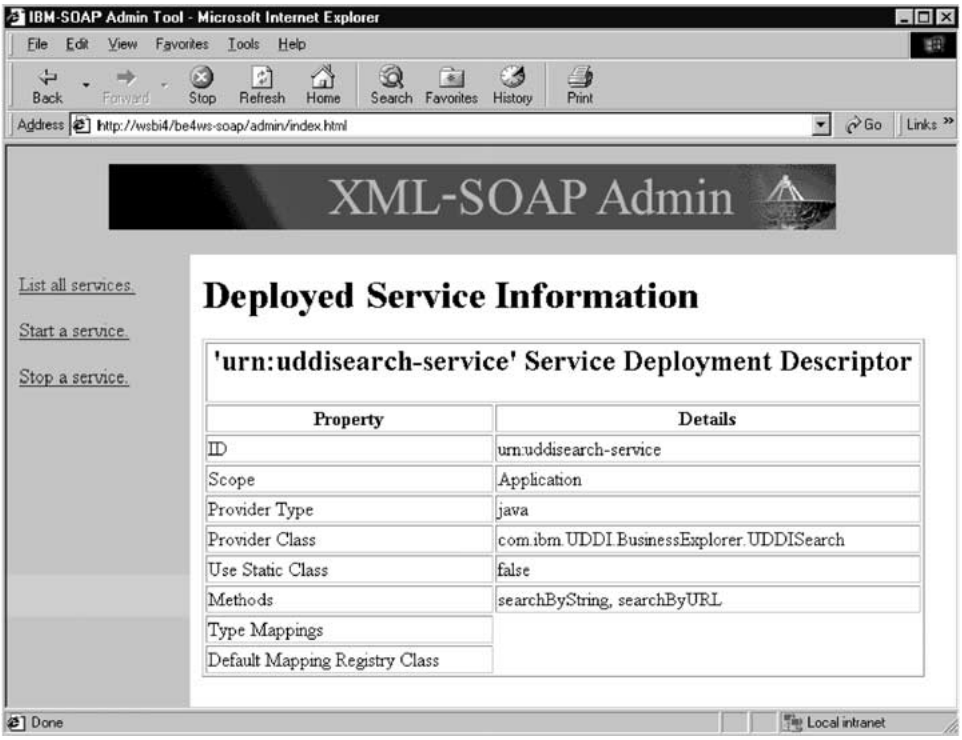

Figure 4. SOAP Service administration console for BE4WS.

6. Sample application of BE4WS for B2B integration

In this section, a working system is presented for demonstrating the feasibility of using the proposed advanced UDDI search engine, BE4WS, for dynamic e-business integration. In this system, buyer can use Web Services to create purchase orders and get the details of the created purchase orders. Meanwhile, suppliers can check the purchase order information, find transportation providers, obtain quotes for service, and assign shipping provider to a specific purchase order. Especially, an intelligent shipping agent making use of BE4WS to search a private UDDI registry is created for filtering at a finer granularity and further narrow down the potential transportation service providers based on the purchase order's country of dispatch.

A sample B2B integration architecture is shown in Figure 5. The PO (Purchase Order) Web Services will be invoked to create PO and get PO details. After a buyer create a PO, it will be stored in the database and processed later. Two purchase orders are listed in Figure 6. The dispatch countries are different, and the shipping agent to locate shipping service providers based on a specific dispatch country for each PO.

Transportation planning is a typical PO processing process. It involves a few flow definitions such as control flows and data flows as well as the business rules that drive the

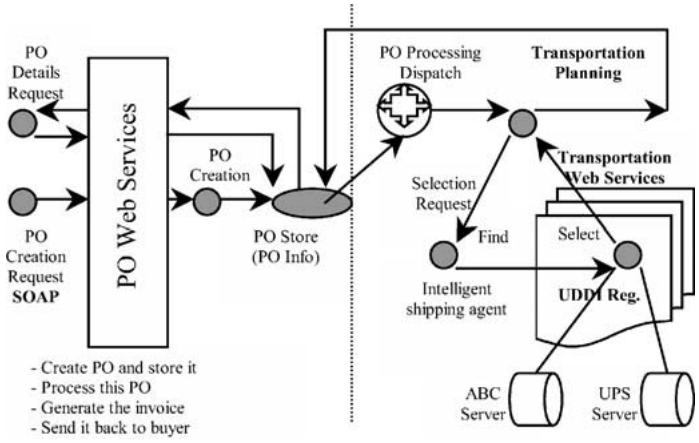

Figure 5. B2B integration architecture.

KEPEX PO Details -> 50-260.1008105319203

Supplier Article Description	Nightstand
Kepex Article Num	4
Supplier Id	999
Customer Name	Metro DE
Qty	80.0
Dispatch Country	UK
Port of Loading	London
E T D	2002-01-10
Transport Service Provider	

Get List of Transporter (E-Hub Memebers Only)

(a)

KEPEX PO Details -> 50-260.1010162303218

Supplier Article Description	Hardwooden Chair
Kepex Article Num	4
Supplier Id	999
Customer Name	Sears
Qty	78.0
Dispatch Country	USA
Port of Loading	New York
E T D	2002-02-10
Transport Service Provider	

Get List of Transporter (E-Hub Memebers Only)

(b)

Figure 6. Purchase order with Dispatch Country: (a) UK; (b) USA.

Table 6. USML script for intelligent shipping agent.

```
<?xml version="1.0"?>
<!DOCTYPE Search SYSTEM "UDDISearch.dtd">
<Search>
  <ProcessId>9999</ProcessId>
  <Query>
    <Source>Private UDDI </Source>
    <SourceURL>http://9.2.168.233:80/services/uddi/inquiryAPI</SourceURL>
    <BusinessName >%trans</BusinessName >
    <FindBy>Business</FindBy>
  </Query>
  <Query>
    <Source>Private UDDI</Source>
    <SourceURL>http://9.2.168.233:80/services/uddi/inquiryAPI</SourceURL>
    <BusinessName >UPS</BusinessName >
    <FindBy>Business</FindBy>
  </Query>
  <AggOperator>OR</AggOperator>
</Search>
```

process choreography. The supplier will select a shipping service provider for this specific PO using a shipping agent, which is designed to search UDDI registry using BE4WS and further filter the search result based on the dispatch country name.

Request from the Web browser will be first forwarded to an access control unit for authentication, which is part of the single-sign-on process. Then the subsequent pages will automatically attach the user credential information to Transportation Planning Process that will invoke a command searching UDDI registry using BE4WS or invoking other Web services based on the business context.

In Figure 6, the PO information is derived from the purchase order database using PO Web Services. The Dispatch Country in Figure 6(a) is UK and the port of loading is London. The Dispatch Country in Figure 6(b) is USA and the port of loading is New York. Based on this information, the intelligent shipping agent will search the UDDI registry to get a transporter list of service providers in UK and USA.

Two levels of search are performed in the shipping service provider selection process: first level involves a human agent defining the search for certain shipping service providers; second level involves using the intelligent shipping agent to automatically narrowing down the preferred list of providers based on the PO contents.

The USML shown in Table 6 is used to look up one private UDDI registry. The human shipping agent wants to get all the service providers which names include "trans" or "UPS". Those two search criteria are created by the shipping agent. Then this USML is used to direct BE4WS to search UDDI registry. Actually, this will result in a set of meaningful service providers who provide shipping service in the dispatch country specified for that particular PO being processed. Next, the intelligent shipping agent processes the result returned by BE4WS and select the certain candidate services to obtain shipping quotes. For the PO shown in the Figure 6(a), the resulting search result from intelligent shipping agent is listed in Figure 7(a), i.e., all transporters in UK. For the PO shown in the Figure 6(b),

Supplier Portal-> Transporters Available List for PO

PO ID:50-260.1008105319203

Transporter:	Description:	Country Served:	Key:	Operator:	Select:
JoeSmith Transportation Inc.	UK. Since its inception in 1977 JoeSmith has been an integral part in the dramatic growth of Mediterranean Shipping Company to its position as 4th largest container shipping company in the world. This has been achieved by total dedication to customers	UK	8375E2F0-E84B-11D5-B61C-970107B90C83	wsbi10/services/uddi	☑
JuneFlower Transportation Inc.	UK. Established in Liverpool in 1989 JuneFlower have developed their business to meet the requirements of a broad cross section of British and Foreign manufacturing industries. JuneFlower have gained a reputation fo Quality, Reliablity and Flexibility.	UK	C9510DD0-E8D8-11D5-B61C-970107B90C83	wsbi10/services/uddi	☑
Lightning Transportation Inc.	UK. Lightning Inc is an independent shipbroking company established in the UK on 1st January 1934. Contact: 220 St Dunstans Hill	UK	AFB44350-E84D-11D5-B61C-970107B90C83	wsbi10/services/uddi	☑

Get Transporter Quotes

(a)

Supplier Portal-> Transporters Available List for PO

PO ID:50-260.1010162303218

Transporter:	Description:	Country Served:	Key:	Operator:	Select:
ABC Transportation Inc.	USA. Corporation provides integrated transportation, information, and logistics solutions through a powerful family of companies that operate independently yet compete collectively U.S. Customer Service: 1-800-Go-ABCTS.	USA	C01923A0-E848-11D5-B61C-970107B90C83	wsbi10/services/uddi	☑
Sunshine Transportation Inc.	USA. Sunshine, the world's largest package distribution company, transports more than 3 billion parcels and documents annually. Corporate Headquarters: 155 Greenwitch Parkway, NE, Atlanta, GA 30328	USA	2BF10F60-E84A-11D5-B61C-970107B90C83	wsbi10/services/uddi	☑
UPS	USA. UPS, or United Parcel Service Inc., is a 93-year-old company whose history spans the bicycle to the Internet. Today the largest express carrier and largest package carrier in the world, UPS continues to develop the frontiers of logistics.	USA	339CFD70-E84D-11D5-B61C-970107B90C83	wsbi10/services/uddi	☑

Get Transporter Quotes

(b)

Figure 7. Service provider list for the PO. (a) Dispatch Country Name is UK; (b) Dispatch Country Name is USA.

the resulting search result from intelligent shipping agent is listed in Figure 7(b), i.e., all transporters in USA. For each service provider, they may provide multiple services. For each service, the supplier can invoke the Web Services to get the quotes for this service. For example, the quotes of services in Figure 6(b) are shown in Figure 8. There are 7 services provided by UPS. The check boxes indicate which transporters that the agent wants to obtain quotes from.

In this example, all the shipping agent has to do in order to obtain the list of shipping Web services that match the dispatch country is to create a script-based USML, and let

Supplier Portal -> Transporter Quotes List for PO

PO ID: 50-260.1010162303218

Transporter	Service Name	Service Invoked	Quote (USD)	Select
ABC Transportation Inc.	ABC_Transport_Service	getTransportationQuote	245.38	○
Sunshine Transportation Inc.	Sunshine_Transport_Service	getTransportationQuote	219.87	○
UPS	Next Day Air, FROM Hawthorne, NY 10532 TO San Francisco, CA 94102 (Your Packaging, 120 pounds)	nextDayAir	308.04	○
UPS	Second Day Air	secondDayAir	206.03	◉
UPS	Third Day Select	thirdDaySelect	144.27	○
UPS	UPS Ground	Ground	58.17	○
UPS	Next Day Air Saver	nextDayAirSaver	275.3	○
UPS	Next Day Air Early AM	nextDayAirEarlyAM	335.75	○
UPS	Second Day Air AM	secondDayAirAM	231.12	○

Accept

Figure 8. Quotes for services.

BE4WS do the magic of searching the private UDDI registry. No nitty-gritty work with the low level UDDI programming.

After the supplier reviews the quotes, he/she can select one as a transportation service provider for this specific purchase order. Then the purchase order information will be updated with the price and considered shipping service provider. The updated purchase order information is shown in Figure 9. At the same time, the notification will be sent to the buyer, supplier and shipping company.

From this working B2B solution integration, we have learned that Web Services are emerging e-business applications that can connect and interact with one another on the Web more easily and efficiently. By using the advanced UDDI search engine, BE4WS, it enables the development of powerful business services, eliminating much of the time-consuming custom coding required by the current UDDI search technology and supporting dynamic and collaborative B2B activities powered by UDDI registry. BE4WS also facilitates dynamic business process brokering and intelligent agents with real-time business services.

KEPEX Supplier Portal PO: Selected Transporter Review

PO ID: 50-260.1010162303218

Kepex Article Num	Hardwooden Chair
Supplier Article No	Hardwooden Chair
Supplier Article Num	4
Supplier Id	999
Cust num	597
Qty	78.0
Dispatch Country	USA
Port of Loading	New York
E T D	2002-02-10
Transport Service Provider	UPS

The PO is finalized. The notification will be sent to the buyer, shipping company and supplier.

Figure 9. Updated PO with assigned service provider.

7. Related work

Web service discovery described in DAML for Web Services (DAML-S) [McIlraith, Son, and H. Zeng, 3] involves the automatic location of Web Services that provide a particular service and that adhere to requested constraints. With DAML-S markup of services, the information necessary for Web service discovery could be specified as computer-interpretable semantic markup at the service Web sites, and a service registry or ontology-enhanced search engine could be used to locate the services automatically. Alternatively, a server could proactively advertise itself in DAML-S with a service registry, so that requesters can find it when they query the registry. Thus, DAML-S is trying to provide declarative advertisements of service properties and capabilities that can be used for automatic service discovery. The disadvantage of DAML-S is it attempts to create a new Web services registry from scratch. If it can describe the semantic information of regular Web services published in UDDI registries or WS-Inspection documents, the DAML-S can be adopted by the next generation Web services discovery mechanism.

Web Service Discovery Architecture (WSDA) promotes an interoperable web service layer by defining appropriate services, interfaces, operations and protocol bindings for discovery [Hoscheck, 2]. The defined interfaces and protocol bindings are very hard to connect to the existing Web services infrastructure such as UDDI registry and SOAP server.

8. Conclusions and future research

With the emergence of interoperable, ubiquitous Web Services published in the UDDI, businesses have a means to describe their services in a global environment and potential

trading partners have a way to discover and interact with each other. It is a critical problem to efficiently find proper Web Services in less time and effort. UDDI Search Markup Language (USML) aids in an efficient search by considering multiple criteria for the desired search from a single or multiple registries. USML is an XML-based language proposed to uniform the search query format and dramatically reduce requesting times in a search. An USML-based search request incorporates multiple search queries, UDDI sources, and aggregation operators. Thus, it takes several criteria in account such as keywords to search for, identifiers, categories and so on for the desired search from a single or multiple registries.

Based on the defined USML, a framework of Advanced UDDI Search Engine (AUSE) is proposed to conduct the searching process. The basic idea of AUSE is to aggregate search results from different UDDI registries based on the proposed USML and its supporting intelligent search facilities such as Instant Notification Broker, UDDI Source Dispatching Broker and Information Aggregation Broker, which have both priori knowledge of the meanings of specific categories and the ability to cross-reference across multiple categories. The Aggregation Broker will parse and re-organize the returned results from different UDDI registries based on aggregation operators and rule-based script. The final result is represented as a XML response to the search requester. The proposed advanced UDDI search engine consists of two types of APIs for the business application developers: regular Java API and Web Services Interface.

Finally, we would like to mention here the advanced UDDI Search Engine has an extensible architecture. It can incorporate not only different UDDI technical layers (e.g., UDDI4J or similar UDDI client package), but also different sources (e.g., UDDI registries, enhanced UDDI registry or other sources such as LDAP, Web and so on). Our work has been published at IBM alphaWorks for public access.

One of the future works is to extend the proposed advanced UDDI search mechanism to support search aggregation from multiple Web Services Inspection (WS-Inspection, WSIL) [Ballinger et al., 1] documents. Meanwhile, the Web Services Relationships Language (WSRL) [Zhang, Chang, and Chao, 10] will be integrated with the proposed search mechanism for improving effectiveness. WSRL describes the relationships about the Web services rather than the requests. The current Web services and UDDI specifications lack the definitions and descriptions of the generic relationships among business entities, business services, and operations. However, these relationships are keys to Web services discovery and composing. For example, the partner relationship among business entities defined in WSRL can help Web services discovery mechanism to effectively locate desired service providers based on the specific search criteria. It is our vision that the next generation Web services discovery mechanism ought to be built on uniformed script-based XML search requests, advanced UDDI search engine for federated UDDIs or WS-Inspection documents with result aggregation, and utilization of semantic information and relationships defined in WSRL for value-added Web services discovery.

Acknowledgments

The authors would like to thank Pooja Yadav and Dr. Haifei Li for their development work on the Business Explorer for Web Services. We also would like to thank David Flaxer and Rama Akkiraju for their contributions on the B2B/Web Services Integration Demo. The original version of this paper is published in the proceedings of IEEE Workshop on E-Commerce and Web-based Information Systems (WECWIS'2002) [Zhang et al., 9]. So many thanks to the organizer of WECWIS'2002.

References

[1] Ballinger, K. et al. (2001). "Web Services Inspection Language (WS-Inspection)." http://www-106.ibm.com/developerworks/webservices/library/ws-wsilspec.html.

[2] Hoscheck, W. (2002). "The Web Service Discovery Architecture." Technical Report, CERN IT Division, European Organization for Nuclear Research.

[3] McIlraith, S.A., T.C. Son, and H. Zeng. (2001). "Semantic Web Services." *IEEE Intelligent Systems* 16(2).

[4] Microsoft UDDI Search. (2001). http://uddi.microsoft.com/search.aspx.

[5] Universal Description, Discovery and Integration (UDDI) 2.0. (2001). UDDI.org, http://uddi.org/specification.html.

[6] UDDI for Java (UDDI4J). uddi4j.org, http://www.uddi4j.org/.

[7] Web Services Description Language. www.w3.org/TR/wsdl.

[8] Web Service Invocation Framework (WSIF). (2001). http://www.alphaworks.ibm.com/tech/wsif.

[9] Zhang, L.-J., H. Li, H. Chang, and T. Chao. (2002). "XML-Based UDDI Search Mechanism for B2B Integration." In *IEEE Workshop on E-Commerce and Web-Based Information System (WECWIS)*, Newport Beach, CA, June 2002, pp. 9–16.

[10] Zhang, L.-J., H. Chang, and T. Chao. (2002). "Web Services Relationships Binding for Dynamic E-Business Integration." In *Internat. Conference on Internet Computing (IC'02)*, Las Vegas, NV, June 24–27, 2002, pp. 561–567.