Discovering Web Services: an overview

Vadim Draluk

BEA Systems, Inc.
2315 North First Street
San Jose, CA 95131
USA
Vadim.Draluk@bea.com

Abstract

The paper introduces a nascent class of specialized database applications, Web service directories (WSD). We discuss changes in computing environment necessitating such applications, and review currently available and emerging products and standards.

1 What's happening in the industry

Oh those simple old days (2-3 years ago) ! Whether used for business or recreational purposes, WWW was then a medium clearly geared towards supporting human interface. While finding things on the 'Net has never been easy, it was implied that human intelligence will always be part of the process, filtering out irrelevant information. So if you had vague search criteria, you would resort to search engines, and sift through the results to find (hopefully) what you were looking for. Otherwise you would probably go directly to one of 'Net hub sites (like Yahoo or Netscape, or one of your professional organisation), and drill down to the subject area of interest.

On the other hand, the enterprises' applications have been exclusively concentrated within the organisation, or, as with EDI, exposed to a small and well-defined set of partners. The most common way to catalogue such applications was using an LDAP-compliant directory, or some other proprietary solution.

Enter Web services and B2B e-commerce. Now WWW becomes a medium to which the need for programmatic access is paramount, and human intervention is not automatically assumed. Now, finding "things" (partners, services etc) has to become much more structured and well defined, so a new class of applications, Web service directories (WSD), has to emerge, supporting this model of interaction. While often dubbed "white/yellow pages" of the Internet, WSDs are significantly different from the garden variety phone books. The latter's work is done once you have found the entry with the number to call: from this point on the protocol is well known (phone conversation), and the information exchange is not structured (at least if you are lucky enough to get a warm body on the other side of the line). Not so with WSD, which, to be useful in establishing programmatic connections, has to go to great length in detailing the description of protocols, interfaces and semantics of services discovered. These applications have to address several common problems:

- Establish a viable and credible business model for themselves
- Define data models to support service publishing and discovery
- Define data access and manipulation mechanisms
- Go global, introducing efficient navigation models between WSD instances
- Choose common ontologies and taxonomies to classify the services
- Establish universal standards to describe the services' syntax and semantics
- Resolve security/trust issues

Currently two competing solutions of these problems appear to be emerging. One, the Universal Description, Discovery and Integration (UDDI) [1, 2, 3, 4], is a registry specification proposed by an industry consortium lead by IBM, Microsoft and Ariba, with more than 250 other companies participating. First announced in September 2000, the effort has already produced working and publicly available prototypes (versions 1 and 2) with functionality reasonably useful at

Proceedings of the 27th VLDB Conference, Roma, Italy, 2001

least for software developers to make practical assessments of the approach.

Another is an OASIS effort, usually referred to as "registry/repository", which incorporates work performed by a separate team as part of the ebXML initiative [6, 7]. This work has been initiated in 1999, with specifications and reference implementations emerging in the first half of 2001. By design it is a somewhat more ambitious effort than UDDI, proposing a general architecture for registering, storing and finding XML documents.

There are many differences in the ways the two approach and resolve (or ignore) the issue listed above, but they also have few things in common. One such aspect is lack of a clearly spelled out business model, particularly strange in case of UDDI, which already offers some relatively robust implementations. This means that, though currently UDDI registries are supported free of charge by the three Operators, it is far from clear how such registries are going to operate in the future. Another feature that the two share conceptually are some previously defined business and geographic taxonomies, such as UN/SPSC, NAICS, ISO 3166 and GeoWeb.

Now let's take a closer look at the technical content of the two efforts, and consider a very simple use case, in which we will try to find all purveyors of organic chemicals in the US (see examples 1–2).

2 UDDI Registry

Data model. Very XML-centric, this simple data model represents *businesses* and their *services*, both classified by externally stored taxonomies. Technical descriptions of services are encapsulated within *binding templates*. All three entities listed above use *keyed references* to the fourth one, called *tModel*. tModels are used to signify classification taxonomies, technical specifications, namespaces etc., and can be considered a generic and powerful mechanism for data model extensions.

Data access and manipulation. Completely XML-based, it relies on SOAP as its transport, and supports APIs for publishing, browsing and drill-down. Data filtering mechanisms are proprietary and very limited, mainly based upon entity names and *tModel* references.

Distributed registries. The basic mechanism provided by the UDDI specification is replication by the so called Operators (currently all three founding members of UDDI). So there's *THE* UDDI registry, represented by many synchronized instances. But one can also register a service description describing a private UDDI registry, supporting the standard (or, may be, extended) UDDI APIs.

Description and classification. Detailed description of services have to be stored outside the registry in form of, mainly, XML documents. UDDI queries never return those documents, only attributes (for *services* and *binding templates*) containing opaque references to them. In practice, for now, they tend not to be that opaque, being URLs of files containing documents. UDDI registries can NOT be queried based on contents of those documents. Recommendations on how to refer from registry to external documents (e.g. WSDL) is described in UDDI "Best Practices" issues [5].

Classification of business, services and binding templates is supported by keyed references to tModels representing well-known taxonomies.

Security and trust. Authentication is provided for data publishing, but not data retrieval by UDDI. At the same time, owners of private registries are encouraged to provide value-add authentication and authorization mechanisms.

There are also capabilities for external data verification, so that third-party services can be used to establish greater level of trust.

The most important feature supporting trust management is called "assertion". An assertion is essentially a relationship instance between two businesses, which becomes visible only if owners of both entries submit it. It enables introduction of "trust" assertions, with hierarchies not unlike those of certificate authorities.

Example 1

```
<find_business xmlns="urn:uddi-org:api"
               generic="1.0" maxRows="100">
   <categoryBag>
     <keyedReference keyName="Organic Chemicals"
         keyValue="1221"
         tModelKey="uuid:DB77450D-9FA8-45D4-"/>
     <keyedReference keyName="United States"
         keyValue="200246"
         tModelKey="uuid:297AAA47-2DE3-4454-"/>
   </categoryBag>
</find_business>
```

This example, of course, makes some critical implicit assumptions:

- That both business and geographic taxonomies are specified at the exact level queried (a business classified with "California" and "Phenols" wouldn't be found)
- That the geographic taxonomy pertains to the area of operation, and not to other things like location of headquarters or of the CEO's skiing chateau.

3 OASIS Registry/Repository

Data model. Some would agree there is none. In fact, there's a meta-meta model, providing for a Registry storing meta-data, and Repository with actual data. Then there is Registry meta-model, with entities such as *Registry Entry, Association, Classification Node, Classification Schema, Package, Slot* etc. There are a couple of very special data model elements, namely *Organization* and *Auditable Event* (so that Registry update log is stored within the registry itself).

The data models are to be hidden in *objectType* attribute of Registry Entry and *associationType* attribute of the Association. So, trying to partially replicate the UDDI data model, we would introduce objectType values of "Business" and "Service", as well as associationType "BusSvc".

The meta-model is expressed in UML, and has a relational mapping, but no XML representation.

Data access and manipulation. Unlike the meta-model, mandatory data access is fully XML-based (with optional SQL implementation). Access is provided to both Registry and Repository data, with fairly complex proprietary XML query language.

Distributed registries. Not well defined, but one could easily imagine a solution similar to one in UDDI. At the same time, OASIS reg/rep instances are expected to be advertised as services in UDDI registries.

Description and classification. There are two approaches to classifications: one is UDDI-like and relies upon *Slots*, a generic extension facility not unlike the UDDI *tModels*. Another is based on storing taxonomies in form of Classification Nodes directly in the Registry.

Detailed descriptions are provided by documents stored in the Repository. There's no search and data filtering based on contents of those documents.

Security and trust. The ACL-based authorization model is described in the specification, but not completely, and as a result is not usable.

Example 2

```
<FilterQuery>
   <RegistryEntryQuery>
      <RegistryEntryFilter>
         objectType EQUAL "Business" AND
         status EQUAL "Approved" AND
         stability NOT_EQUAL "Dynamic"
      </RegistryEntryFilter>
      <SourceAssociationBranch>
      <AssociationFilter>
          associationType EQUAL "BusSvc"
      </AssociationFilter>
      <RegistryEntryQuery>
         <RegistryEntryFilter>
            objectType EQUAL "Service" AND
            status EQUAL "Approved" AND
            stability NOT_EQUAL "Dynamic"
         </RegistryEntryFilter>
      </RegistryEntryQuery>
      <HasClassificationBranch>
         <ClassificationNodeFilter>
            path STARTSWITH "UNSPSC" AND
            code EQUAL "1221"
         </ClassificationNodeFilter>
      </HasClassificationBranch>
     <HasClassificationBranch>
         <ClassificationNodeFilter>
            path STARTSWITH "GeoWeb" AND
            code EQUAL "200246"
         </ClassificationNodeFilter>
      </HasClassificationBranch>
</FilterQuery>
```

The results of this example are slightly different from one in UDDI, due to differences in semantics: this query will return elements classified by *ANY* taxonomy nodes down the classification tree from ones specified in the query.

4 What's next?

Currently UDDI is developing its Version 3 offering, after which it is expected to be turned over to an open standards body. OASIS reg/rep committee is working to reconcile its initial proposal with the one coming from ebXML.

In order to support real-life applications, WSDs have to develop further and mature. More specifically, in terms of WSD functionality, it means:

- Closer integration between Registry and Repository, with querying capabilities extending into the latter
- Repository-oriented functionality, such as versioning
- Better support for security and trust management
- More flexible querying capabilities, based on existing and emerging XML standards whenever possible
- Richer taxonomies, with support of context and semantics

Beyond purely technical issues, there's a vast effort ahead to develop and promote viable taxonomies describing all areas of e-business.

References

[1] UDDI Version 2.0 Data Structure Reference. UDDI Candidate Draft Specification, 28 February 2001

[2] UDDI Version 2.0 API Specification. UDDI Working Draft Specification

[3] UDDI Version 2.0 Operator's Specification. UDDI Candidate Draft Specification, 19 April 2001

[4] UDDI Version 2.0 Replication Specification. UDDI Candidate Draft Specification, 29 May 2001

[5] Using WSDL in a UDDI Registry 1.02. UDDI Working Draft Best Practices Document, February 19, 2001

[6] ebXML Registry Information Model v1.0. ebXML Registry Project Team, 8 May 2001

[7] ebXML Registry Services Specification v1.0. ebXML Registry Project Team, 10 May 2001